CodeWarrior®

Constructor for Java

Because of last-minute changes to CodeWarrior, some of the information in this manual may be inaccurate. Please read the Release Notes on the CodeWarrior CD for the latest up-to-date information.

Revised: 970403-JJGMS

Metrowerks CodeWarrior copyright ©1993–1998 by Metrowerks Inc. and its licensors. All rights reserved.

Documentation stored on the compact disk(s) may be printed by licensee for personal use. Except for the foregoing, no part of this documentation may be reproduced or transmitted in any form by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from Metrowerks Inc.

Metrowerks, the Metrowerks logo, CodeWarrior, and Software at Work are registered trademarks of Metrowerks Inc. PowerPlant and PowerPlant Constructor are trademarks of Metrowerks Inc.

All other trademarks and registered trademarks are the property of their respective owners.

ALL SOFTWARE AND DOCUMENTATION ON THE COMPACT DISK(S) ARE SUBJECT TO THE LICENSE AGREEMENT IN THE CD BOOKLET.

How to Contact Metrowerks:

U.S.A. and international	Metrowerks Corporation 9801 Metric, Suite #100 Austin, TX 78758 U.S.A.
Canada	Metrowerks Inc. 1500 du College, Suite 300 Ville St-Laurent, QC Canada H4L 5G6
Mail order	Voice: (800) 377–5416 Fax: (512) 873–4901
World Wide Web	`http://www.metrowerks.com`
Registration information	`register@metrowerks.com`
Technical support	`support@metrowerks.com`
Sales, marketing, & licensing	`sales@metrowerks.com`
CompuServe	goto `Metrowerks`

Table of Contents

1 Introduction 7

Introduction Overview 7
System Requirements 7
Installing Constructor 9
Starting Points 9
Where to Learn More 10
How is Java for PowerPlant Different? 10

2 Fundamentals 13

Constructor Fundamentals Overview 13
What Constructor for Java Can Do 13
Using Constructor Files 14
Constructor project file 14
Java source file 14
Creating a Constructor Project 16
Managing Constructor project files 17
The Project Window 18
Editor Windows 19
User Interface Features 21
Selection 21
Moving, Copying, and Deleting 22
Lists 24
Text Editing 24
Undo 25
Using the Layout in your Java Program 25
Using the Reanimator class 25
Reanimator function reference 27

3 Managing Components 31

Managing Components Overview 31
Making a New Component 31
Making Any New Component 32
Making a New Layout 33
Creating a Menu Bar 35

Creating a Menu 35
Copying a Component 35
Modifying The Name 36
Opening a Component for Editing 36
Deleting a Component 37

4 Editing Layouts. 39
Editing Layouts Overview 39
Windows for Layouts 40
Catalog window 40
Layout Window 42
Hierarchy Window 43
Layout and Hierarchy Compared. 45
Property Inspector Window 45
Building a Visual Interface 47
Adding Components to a Layout 47
Displaying Component Features 49
Arranging Components Visually 52
Arranging Components Hierarchically 63
Setting Component Properties 65
Removing Components 66
Creating Customized Components 67

5 Editing Menu Bars and Menus 69
Editing Menu Bars and Menus Overview 69
Windows for Menus 69
Menu Bar Editor Window 70
Menu Editor Window. 71
Menu Property Window. 72
Menu Item Property Window 72
Editing a Menu Bar 73
Making a New Menu 74
Adding an Existing Menu 74
Modifying a Menu 76
Changing Menu Order 76
Editing Menu Items 77

Selecting a Menu Item 78
Making a New Menu Item 79
Modifying a Menu Item 80
Making a Menu Separator Item 80
Moving a Menu Item 81
Adding a Submenu 82
Removing a Menu Item 82

6 Component Properties 83

Component Properties Overview 83
Using a Property Inspector Window 84
Anatomy of a Property Inspector 84
Component 87
Layouts 89
Container 90
Panel 90
Window 91
Frame 91
Dialog 92
Controls 93
Button 93
Canvas 94
Checkbox 94
Choice 95
Label 96
List 96
Scrollbar 97
TextComponent 99
TextArea 99
TextField 100

7 Constructor Menu Reference. 103

Constructor Menu Reference Overview 103
Apple Menu 103
File Menu 104
Edit Menu 105

Arrange Menu 107
Layout Menu 109
Window Menu 111

Index 113

1 Introduction

Welcome to the CodeWarrior Constructor for Java manual. Constructor for Java is the visual interface builder for the Java Abstract Windowing Toolkit, or AWT. AWT is a package for creating a graphical user interface and is a standard part of Java.

Introduction Overview

Before using Constructor, read the release notes. They contain important information about new features, bug fixes, and any late-breaking changes.

The other topics in this introduction are:

- System Requirements—hardware and software requirements
- Installing Constructor—putting it together
- Starting Points—an overview of the chapters in this manual
- Where to Learn More—other sources of Constructor information
- How is Java for PowerPlant Different?—how Constructor for Java and Constructor for PowerPlant differ.

System Requirements

If you can run CodeWarrior, you can run Constructor. Constructor requires a Motorola 68020 processor or better, or a PowerPC 601 or better processor. Constructor for Java needs approximately 3.5 MB of RAM (more on a PowerPC with virtual memory turned off): 2.5 MB for Constructor itself and 1 MB for the Java Virtual Machine.

Constructor requires System 7.5 or later, or System 7.1 with the Thread Manager and the Drag and Drop extensions installed. If

you're using a 68K Macintosh, you also need the CFM68K extension.

Constructor for Java is part of a complete package that also includes the CodeWarrior IDE, Metrowerks Java, and the Mac OS. You need all these elements to use Constructor for Java effectively.

Menu Editor

Constructor for Java contains a new menu editor for creating Java menus. See "Editing Menu Bars and Menus Overview."

Customized Components

You can use your own customized versions of the standard AWT components in your interface. See "Creating Customized Components."

Specifying a package

You can now specify the package for the code that Constructor generates. See "Java source file."

New Catalog Window

The Display Classes Window that lists all components for use in editing layouts has been replaced. The new window is called the Catalog Window. It has the same functionality as the former Display Classes window, but now organizes components into logical groups so you can find what you're looking for more easily. The interface is based on the same design used in the Debugger's preference panel.

This design also allows for future expansion of the catalog to include other items besides components.

New Property Inspector Window

The Property Inspector window now identifies the component's type and name at the top of the window. See "Using a Property Inspector Window."

Installing Constructor

Installing Constructor is very simple. Locate the Constructor for Java folder on the CodeWarrior Tools CD. Copy the folder to your hard drive. Then, put `Reanimator.zip` in the Classes subfolder of the Java folder in your System folder.

Starting Points

This manual consists of the following chapters:

- Introduction Overview—this chapter
- Constructor Fundamentals Overview—an overview of Constructor, including a discussion of the Constructor project window
- Managing Components Overview—creating, copying, modifying, and deleting components in a Constructor project
- Editing Layouts Overview—creating and editing a visual interface
- Editing Menu Bars and Menus Overview—creating and editing Java menus
- Component Properties Overview—a reference to each pane property inspector
- Constructor Menu Reference Overview—a reference to each Constructor menu item

If you have never used Constructor, then you should begin with "Constructor Fundamentals Overview" before proceeding to other topics. This chapter covers the various elements of the Constructor interface, and how they work together.

"Managing Components Overview," is also excellent for new Constructor users. It discusses how to create new components.

Use the remaining chapters in this manual as reference material. They discuss how to edit components of various types.

This manual is a user's guide to Constructor. When you have a question about the interface, or about how to accomplish a particular task, refer to the chapter that covers the topic in which you are interested.

Where to Learn More

CodeWarrior contains a good deal of information about Java and the Abstract Windowing Toolkit (AWT). *Learn Java on the Macintosh* teaches you how to use Java and contains a chapter with basic information on creating a user interface. And the *Java API Documentation* is a class-by-class description of everything in the Java API, including the AWT.

How is Java for PowerPlant Different?

Constructor for Java is a direct descendent of Constructor for PowerPlant. If you know how to use one, you should be able to use the other very easily. However, there are some differences you need to be aware of.

Strings vs. four character constants Constructor for PowerPlant uses four character constants for identifying layouts and their components. Constructor for Java uses strings.

Resources vs. Source Constructor for PowerPlant creates resources which are linked into an application and parsed at runtime by the UReanimator class. Since Java has no equivalent to resources, Constructor for Java cannot work this way. Instead, Constructor for Java generates Java source code that must be included in your project. However, Constructor for Java does store the layout data as resources to avoid adding a Java parser to Constructor and avoid the possibility of error if you edit the source code. (Constructor for Java *never* parses source).

Reanimation process -There is no standard mechanism in the AWT for constructing components from files. Since Java is a single inheritance language, there's no easy way to add this feature easily. Therefore, the Java Reanimator class instantiates an instance of the

class you're creating, and then invokes methods in order to set properties.

Custom classes In Constructor for PowerPlant, you can add custom pane types to the catalog. In Constructor for Java, you cannot add custom component types, but you can replace the components it already handles with customized versions. For more information, see "Creating Customized Components."

2

Fundamentals

This chapter is an introduction to Constructor for Java. It gives you a high-level overview of what Constructor is and how to use it effectively.

Constructor Fundamentals Overview

Constructor for Java is a graphical editor designed to create and edit a user interface for Java programs. Constructor was originally created as a resource editor to accompany PowerPlant. As such, it has become the premier visual interface builder for the Macintosh.

The current version of Constructor for Java lets you edit containers, such as Frames, Panels, and Dialogs, and the components they contain, such as Buttons, Checkboxes, and TextFields.

The topics in this overview include:

- What Constructor for Java Can Do—component types that Constructor understands
- Using Constructor Files—the files Constructor works with
- The Project Window—the display for a project file
- Editor Windows—how individual component editors relate to the project window
- User Interface Features—a quick look at the user interface
- Using the Layout in your Java Program—how to use the interface in your Java program.

What Constructor for Java Can Do

Fundamentally, Constructor for Java lets you create and edit a user interface for a Java program that uses the Abstract Windowing Toolkit. The classes that Constructor manages include:

- Containers—a panel or window that can contain user interface components.
- Components—user interface elements such as buttons, checkboxes, and text fields.

Whenever you save a Constructor project file, Constructor automatically creates a Java file that uses the information in your project file to create a layout for your program. To use a layout you created in Constructor, just call the `Reanimator.Reanimate` method specifying the layout's class and the name you assigned that particular layout. For example, if you create a frame named `MyFrame`, you would use it in your program with the statement:

```
Reanimator.Reanimate("Frame","MyFrame");
```

Using Constructor Files

Constructor for Java maintains two files for your Constructor Java: a Constructor project file and a Java source file:

Constructor project file

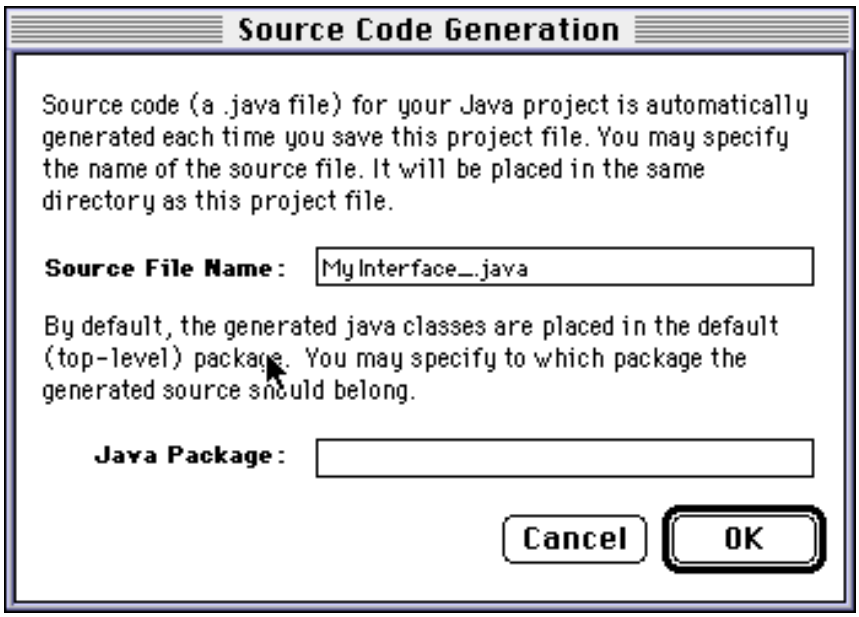
Constructor uses this file to store information about your user interface. By convention, Constructor for Java project file names end with the extension `.jpob`, since information about Java containers and their components is stored in `'JPob'` resources in the project file.

Java source file

Your Java program uses the code in this file when you want to use the interface you just created in Constructor. Constructor creates this file for you automatically whenever you save the project file. By default, Constructor gives this file the same name as the project file, except that it ends in `_.java` instead of `.jpob`. For example, if you named the project file `MyInterface.jpob`, the Java file is named `MyInterface_.java`.

If you want to change the name of the Java file or specify a package for the generated code, choose **Generate Source…** from the **File** menu. It displays the following dialog:

Figure 2.1 Generate Source dialog


Source Code Generation
Source code (a .java file) for your Java project is automatically generated each time you save this project file. You may specify the name of the source file. It will be placed in the same directory as this project file.
Source File Name:
MyInterface_.java
By default, the generated java classes are placed in the default (top-level) package. You may specify to which package the generated source should belong.
Java Package:
Cancel
OK

In the **Source File Name** field, enter the name of the Java file.

In the **Java Package** field, enter a package name. When Constructor generates the source, it adds a package statement at the beginning of the file with that name. Note that when you use the interface you created in your Java program, you must specify this package in the routines that re-create your interface. For more information, see "Using the Layout in your Java Program."

WARNING! Do not edit the Java source file that Constructor generates. The next time you edit and save the Constructor project file, Constructor will write over the Java source file and erase your changes.

Creating a Constructor Project

1. **Choose New Project from the File menu.**

   Constructor creates a new project file and displays its window.

2. **Create your user interface.**

   Use the techniques discussed in the rest of this manual.

3. **Choose Save As… from the File menu.**

   Constructor displays a dialog asking you to name the file. Constructor then saves two different files:

   - A project file that Constructor uses when you want to edit your interface.
   - A Java source file which contains code your Java program uses when you want to recreate the interface you just created in a Java program.

4. **Add the Java file and Constructor project file to your CodeWarrior project.**

   You must add the Java file to your project. If you don't, you cannot use your user interface in your Java program.

   You don't need to include the project file, but you'll find it convenient. You'll have all the files for your project together in one window, and you'll be able to open the Constructor file directly from CodeWarrior by double-clicking it in the project window.

   To add the files, you can either use the **Add Files…** command in CodeWarrior, or you can drag the files directly from the Finder into your CodeWarrior project.

5. **Add the Reanimator.zip file to your CodeWarrior project.**

   This library contains the routines that Java source file uses to recreate your user interface.

   Also make sure your project contains the `Classes.zip` file, which all Java projects need.

6. **In your Java program, use the Reanimator.Reanimate() function to recreate your user interface.**

   For more information, see "Using the Layout in your Java Program."

7. **Continue to edit and save your Constructor project file.**

   Each time you save the Constructor project file, Constructor automatically generates a new Java file that recreates that interface.

Managing Constructor project files

Use these **File** menu commands to manage project files.

Table 2.1 Managing a Constructor project file

To…	Do this
Create a new file	Choose **New Project File** in the **File** menu.
Open a file	Choose **Open Project File** in the **File** menu, or double-click the file in the Finder.
Save a file	Choose **Save** in the **File** menu. Constructor also generates a new Java file for the project.
Save a file with a new name	Choose **Save As…** in the **File** menu. Constructor also generates a new Java file for the project.
Change the name of Java file.	Choose **Generate Source** from the File menu, and enter the new name in the dialog that appears. Constructor generates a new Java file for the project, without saving the project file.
Close a file	Choose **Close** in the **File** menu, or click in the close box of the project window.

Your ability to save changes will be affected by the status of the file. If the file is locked or read-only, you can't save any changes. If the

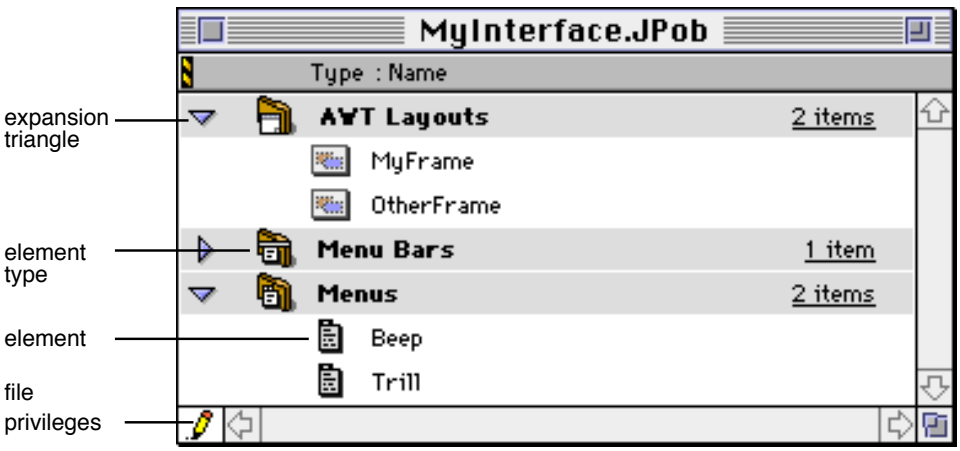
file has modified read-only status from a source code control system, you may not be able to check the file back into the system.

For more information on this topic, see the CodeWarrior IDE User Guide section on file privileges.

The Project Window

When you open a Constructor project file, Constructor displays a window that lists each type of component, and the individual components available for each type. Figure 2.2 shows this window.

Figure 2.2 A Constructor project window



The project window is the central or main Constructor window where you create new components and manage existing components. The "Managing Components Overview" chapter covers this topic in detail.

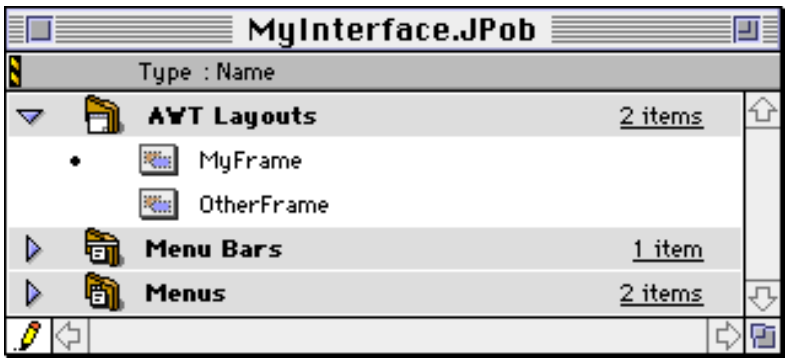
Each grey bar in the project window represents one type of user interface element. There are three types:

- Layouts
- Menu bars
- Menus

To the left of each type heading is an expansion triangle. Open or close the list of elements of that type by clicking on the triangle.

Whenever any layout listed in the project window has changed and has not been saved, a bullet appears to the left of that component, as shown in Figure 2.3, where the MyPanel layout has changed.

Figure 2.3 A changed component



You may have multiple projects open simultaneously, and copy components from one to another.

The file privileges icon may appear in the lower left corner of the project window to indicates the read/write status of the file, usually with respect to a source code control system. The file privileges icon works just like the same icon in the CodeWarrior IDE. For information on this topic, see the IDE User Guide section on file privileges.

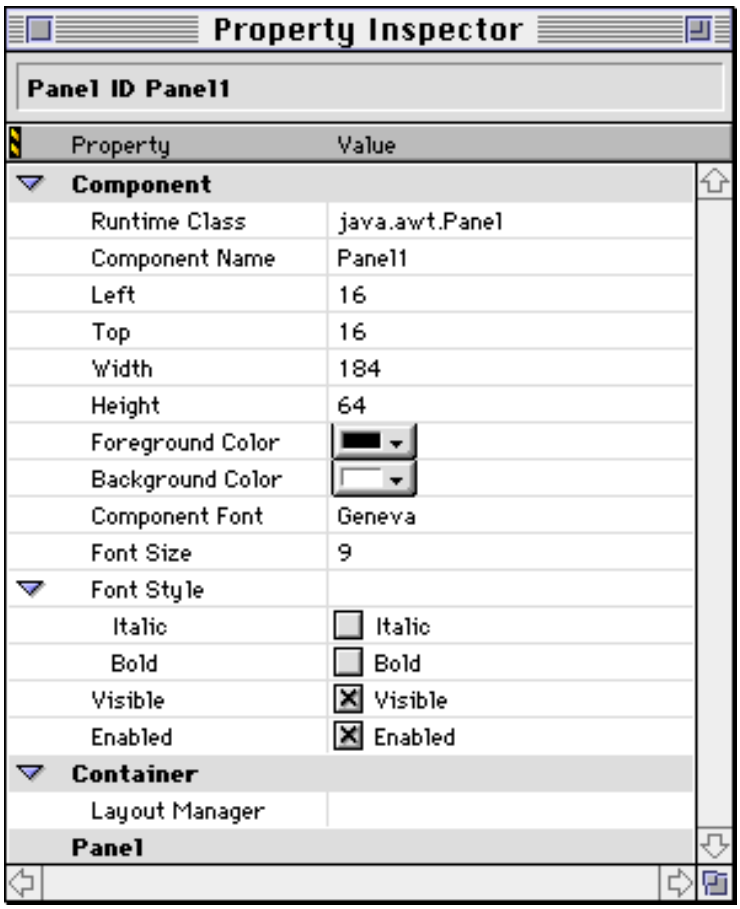
Editor Windows

To open a component for editing, use one of the following techniques:

- double-click it in the project window
- select it and press the Return or Enter key
- select it and choose **Edit Layout…** from the **Edit** menu, if it's a layout.

Whichever you choose, Constructor opens an editor window that allows you to modify the component. Each kind of component has a different editor window. Figure 2.4 shows the Panel editor window.

Figure 2.4 The Panel editor window



The chapter "Editing Layouts Overview" discusses the individual editors in detail.

User Interface Features

Constructor has many of the interface features you expect in a good Mac OS program. These include features that support:

- Selection—selecting items
- Moving, Copying, and Deleting—relocating and copying items
- Lists—managing lists of items
- Text Editing—modifying text in an item
- Undo—support for undo and redo

The figures in this section use the project window to illustrate the interface. The same techniques work in individual editor windows, when the technique is appropriate for the kind of data displayed in the window. Interface features are not available in all windows under all circumstances. In some situations, particular features are not appropriate.

Selection

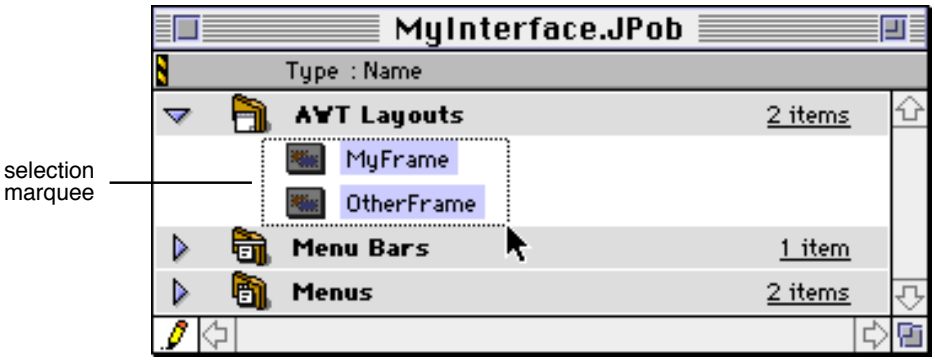
Constructor follows the standard human interface behavior with respect to selecting items. In general, you first select an item or group of items, then you perform an operation on the item(s).

To select an item, you click it.

To extend a selection (select an additional item while keeping previously selected items), you press the Shift key and click.

To select contiguous items with one gesture, click and drag the mouse. Items that intersect the area defined by the resulting selection marquee become selected when you release the mouse button. Figure 2.5 illustrates a selection marquee.

Figure 2.5 Selecting multiple items with a marquee



For marquee selection to work, the drag must begin in an empty area of the window. If you click and drag on an item, you move or copy the item instead of starting a marquee.

TIP: For a layout window, a drag that begins in the top-level view and not inside an individual pane starts a marquee selection. If you start your drag inside an individual Panel, you select the panel and then move or resize it. To get around this problem, hold down the Control key when you being a drag inside a panel.

To select all items in a window, choose the **Select All** item (Command-A) in the **Edit** menu.

To deselect all selected items, click outside of any item. For example, you can click the simulated desktop in a layout window to deselect all selected items. In the layout window only, you may also press the Control key while clicking an item. See "Layout Window."

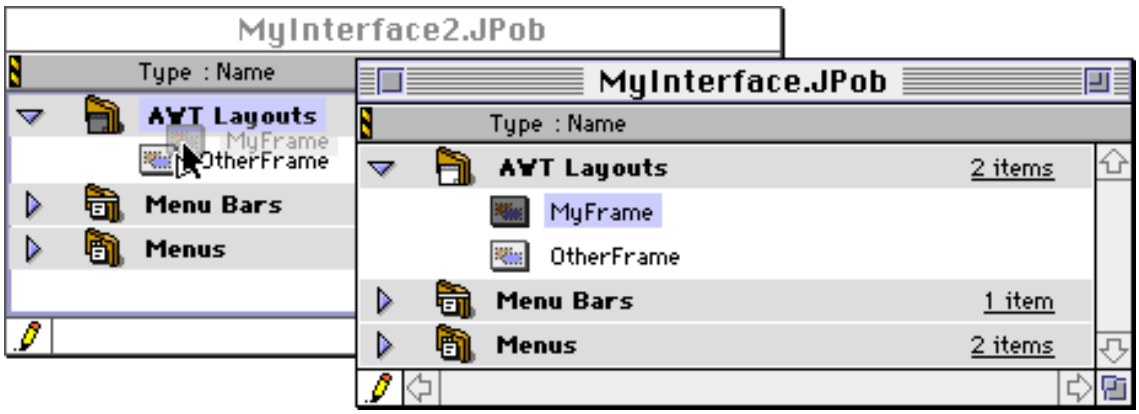
Moving, Copying, and Deleting

To move or copy an item, you click and drag it.

If you drag an item to another location in the same window, you move it.

If you drag an item from one window and drop the item in another window, Constructor makes a copy of the item in the destination window. The item is *not* removed from the originating window. Figure 2.6 illustrates copying a component by dragging from one project window into another.

Figure 2.6 Copying a component by dragging



To copy an item you move within a window, press the Option key when beginning the drag or when dropping the item.

NOTE: Option-dragging to create a copy does not work in the project window. Use the **Duplicate Object** command (see below).

In addition to drag and drop, the standard **Cut**, **Copy**, and **Paste** commands in the **Edit** menu work in the traditional manner at all appropriate times.

You may also choose the **Duplicate Object** command (Command-D) in the **Edit** menu to make a copy of an item in the same window.

To delete an item, select it. Then press the Delete key, or use the **Clear** command from the **Edit** menu.

TIP: When copying, duplicating, moving, or deleting a pane that contains other panes, you work on the container and its contents.

Lists

Some Constructor windows, like the project window, display lists of information. You can navigate through lists of items by typing the up or down arrow keys. Each time you do, the next or previous item in the list becomes selected.

Constructor uses expansion triangles to open and close lists. Click on the triangle to change the state of the associated list. If the triangle is next to a selected item, you may also type the right or left arrow keys to open or close a list, respectively.

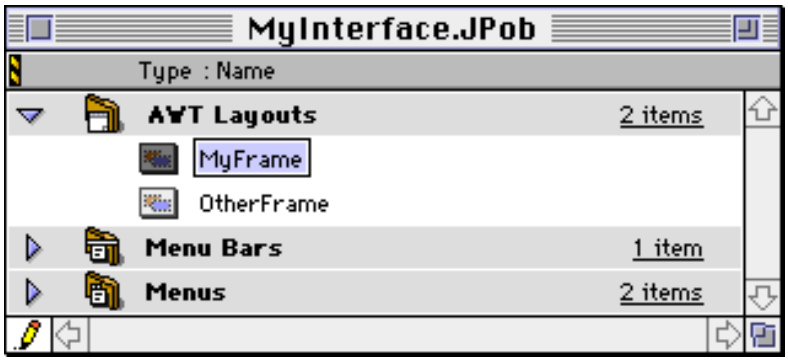
You can extend a selection in a list by pressing the Shift key and typing the up or down arrow key.

Text Editing

Textual items appear in many places in Constructor windows. For example, lists of items have names. Many windows have editable text fields.

To activate a text item for editing, click on the item. Then either move the mouse off the text, or wait a second (exactly as you would in the Finder). You know the item is editable when a frame appears around the text, the text in the item becomes selected, or when the text insertion cursor appears in the item. Figure 2.7 shows how you can edit items in the project window.

Figure 2.7 Editing text in-place in the project window.



You navigate through the text in an item by using the arrow keys. You select text by dragging with the mouse in the text item.

Undo

Constructor has single-level undo. Most operations can be undone (and subsequently redone) using the **Undo** or **Redo** items in the **Edit** menu (Command-Z).

If undo is unavailable after a given action, the menu item will read **Can't Undo** .

Using the Layout in your Java Program

This section describes how the Reanimator class lets you use the interface you created in Constructor.

- Using the Reanimator class
- Reanimator function reference

Using the Reanimator class

After you create a user interface in Constructor for Java, you need to modify your Java program to use it. Usually, all you'll need to do is add one line of code that calls the `Reanimator.Reanimate()` function, described in "Reanimate." It "reanimates" your interface,

including the top-level layout and all its components, and returns the top-level layout. After you call this function, you can use your layout like you would any other Java AWT layout.

For example, this listing reanimates the Panel with the Component Name `TitlePanel`.

Listing 2.1 Reanimating a layout

```
Panel p1 = (Panel) Reanimator.Reanimate("Panel", "MyPanel");
```

If you specified in the Generate Source dialog that the constructor's generated source should go into a specific package, you need to specify that package, like this:

Listing 2.2 Reanimating a layout in a package

```
Panel p1 = (Panel) Reanimator.Reanimate("Panel", "MyPanel",
                                        "MyPackage");
```

Note that if the source was generated into a specific package, you need to specify that package with any of the Reanimate functions, described below.

If you're reanimating a dialog or window, you need to call a different function, `Reanimator.ReanimateWindow()`, and specify the main application frame. It's described in "ReanimateWindow."

You can also find a component within a layout. The function `Reanimator.Locate()`, described in "Locate," returns a component you specify. For example, this listing returns the TextField named `UserName` that's in the layout created above, `panel`:

Listing 2.3 Locating a component

```
TextField t1 = Reanimator.Locate("TextField", "UserName", panel);
```

If you used Constructor to create menus, you'll usually use `Reanimator.ReanimateMenuBar()` to reanimate a menubar and

all its menus. It's described in "ReanimateMenuBar." To reanimate a single menu, use `Reanimator.ReanimateMenu()`. It's described in "ReanimateMenu."

Reanimator function reference

The following sections describe all the functions in the Reanimator class. Each section is named after the function, and contains the functions declaration, a short description, and a table of its parameters. In the declaration, optional parameters appear in brackets.

- Reanimate
- ReanimateWindow
- ReanimateMenu
- ReanimateMenuBar
- Locate

Reanimate

```
public static Component Reanimate (String
inClassName,
     String inObjectName[, String inPackageName]
)
```

Reanimate a component.

Table 2.2 Reanimate parameters

`inClassName`	The name of the class of the component. This must be fully qualified; for example, use `"java.awt.Frame"` instead of `"Frame"`.
`inObjectName`	The name of the component, as specified in Constructor in the **Component Name** field
`inPackageName` (Optional)	The package that Constructor's generated code is in. You specify this package in the Generate Source dialog. If you didn't specify a package, don't include this parameter.

ReanimateWindow

```
public static Window ReanimateWindow(
    String inClassName, String inObjectName,
    Frame inParent[, String inPackageName] )
```

Reanimate a window or dialog. Since you need to specify a parent frame when creating a window or dialog, you cannot use the general-purpose Reanimate function. However, to reanimate a frame, use the general-purpose `Reanimate` function.

Table 2.3 ReanimateWindow parameters

`inClassName`	The name of the window subclass. This must be fully qualified; for example, use `"java.awt.Window"` instead of `"Window"`.
`inObjectName`	The name of the window object, as specified in Constructor in the **Component Name** field.
`inParent`	The main application frame.
`inPackageName` (Optional)	The package that Constructor's generated code is in. You specify this package in the Generate Source dialog. If you didn't specify a package, don't include this parameter.

ReanimateMenu

```
public static Menu ReanimateMenu(
     String inMenuName[, String inPackageName])
```

Reanimate a Menu.

Table 2.4 ReanimateMenu parameters

`inMenuName`	The name of the Menu.
`inPackageName` (Optional)	The package that Constructor's generated code is in. You specify this package in the Generate Source dialog. If you didn't specify a package, don't include this parameter.

ReanimateMenuBar

```
public static MenuBar ReanimateMenuBar(
    String inBarName[, String inPackageName])
```

Reanimate a MenuBar and all the menus it includes.

Table 2.5 ReanimateMenu parameters

`inBarName`	The name of the menubar.
`inPackageName` (Optional)	The package that Constructor's generated code is in. You specify this package in the Generate Source dialog. If you didn't specify a package, don't include this parameter.

Locate

```
public static Component Locate(
    String inClassName, String inObjectName,
    Container inRoot[, String inPackageName])
```

Locate an object in a reanimated hierarchy. This method returns a reference to a component in a Constructor hierarchy at runtime.

Table 2.6 Locate parameters

`inClassName`	The name of the component's class. This must be fully qualified; for example, use `"java.awt.Button"` instead of `"Button"`.
`inObjectName`	The name of the component object, as specified in Constructor in the **Component Name** field
`inRoot`	The container returned from the `Reanimate()` method: the top level container of the layout
`inPackageName` (Optional)	The package that Constructor's generated code is in. You specify this package in the Generate Source dialog. If you didn't specify a package, don't include this parameter.

3

Managing Components

This chapter discusses how to create, copy, modify, open, and delete components in a Constructor project.

Managing Components Overview

You manage components from the Constructor project window. For details on the project window, see "The Project Window." In general, you select the component or components you want to work with, and then do something with or to that component.

The topics discussed in this chapter include:

- Making a New Component
- Copying a Component
- Modifying The Name
- Opening a Component for Editing
- Deleting a Component

For the most part, you perform each of these tasks in the same way regardless of the type of component involved.

Making a New Component

To make a new component, choose **New Resource** from the **Edit** menu. What happens next depends upon the selection in the project window.

Based on the selection, if Constructor knows for sure what kind of component you want, it creates an untitled component of the correct

type. If Constructor does not know what type of component you want, Constructor displays a dialog that asks you to specify additional information.

Other chapters in this manual discuss how to edit the contents of each individual type of component. This section simply discusses how to work with the component as a whole.

The topics in this section are:

- Making Any New Component
- Making a New Layout
- Creating a Menu Bar
- Creating a Menu

Making Any New Component

In the project window, if you have no component selected, a layout selected, or components of different types selected, then Constructor does not know for sure what kind of component you want to create. Constructor presents the Create New Resource dialog, as shown in Figure 3.1.

TIP: To see this dialog no matter what component (if any) is selected, use Command-Option-K.

Figure 3.1 The Create New Resource dialog

You select the desired component type from the Resource Type pop-up menu, as shown in Figure 3.2.

Figure 3.2 Available component types

If the resource type is a layout, you select the top-level container type from the Kind pop-up menu. If the resource type is not a layout, the Kind pop-up menu is disabled. You may also specify the component name. Then click the Create button to create the new component.

Use this technique to make a Constructor component of any type. Each type also has one or more methods for creating that specific type.

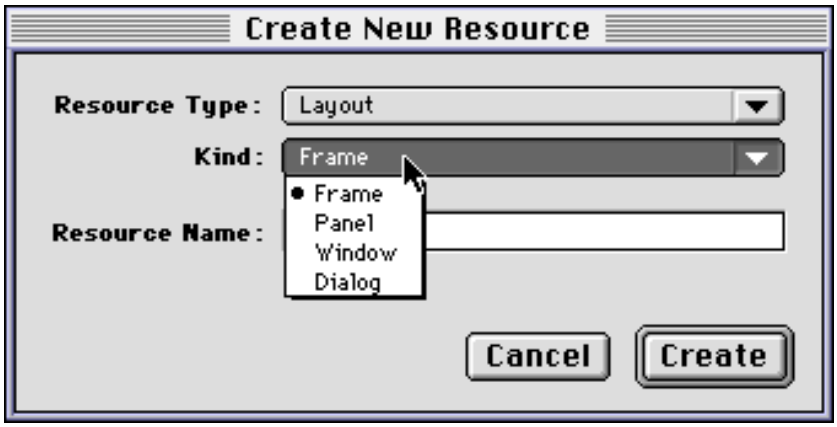
Making a New Layout

There are three ways to create a new layout.

- Follow the instructions in Making Any New Component.
- Select an existing layout in the project window (or the grey resource type header bar), and then choose **New Resource** from the **Edit** menu.
- Drag a layout from the Catalog window.

If you have a layout selected in the project window, when you choose **New Resource** the Create New Resource dialog illustrated in Figure 3.3 appears. You must also choose the type you wish to be the top-level type for this layout.

Figure 3.3 The New Layout dialog



You may choose from Frame, Panel, Window, or Dialog. Choose the top-level type you wish this layout to have, and click the **Create** button.

Table 3.1 Choosing the top level type in a layout

Kind	Use for
Frame	A top-level window with a title, a border, and, optionally, a menu bar.
Panel	Applets or grouping together related components inside another layout.
Window	A top-level window with no title, no border, and no menu bar. Although Constructor for Java lets you use a Window as a layout, you'll usually use one of its subclasses instead: Frame or Dialog.
Dialog	A dialog window.

Creating a Menu Bar

There are two ways to create a new menu bar.

- Follow the instructions in Making Any New Component.
- Select an existing menu bar in the project window (or the grey resource header bar), and then choose **New Resource** from the **Edit** menu.

Constructor creates a new, untitled menu bar.

Creating a Menu

There are two ways to create a new menu.

- Follow the instructions in Making Any New Component.
- Select an existing menu in the project window (or the grey header bar), and then choose **New Resource** from the **Edit** menu.

Constructor creates a new, untitled menu.

Copying a Component

When you create a new component, you must then edit the component to define its contents properly. (How to edit individual components is discussed elsewhere in this manual.) Sometimes it is more efficient to make a copy of an existing component, and then make minor modifications to the copy.

There are three ways to copy a component. First, you select one or more components. Then you can:

- use the **Copy** and **Paste** commands in the **Edit** menu
- use the **Duplicate Object** command in the **Edit** menu
- drag the component(s) into another project

WARNING! You must change the name of the duplicate, if you make a copy in the same project, or if the destination project has a

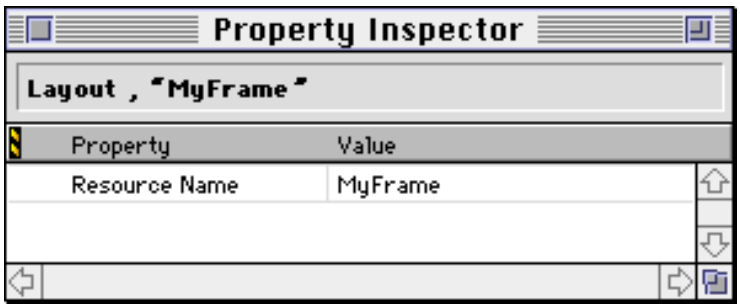
component of the same class with the same Component Name. The Reanimator code does not work correctly if there are duplicate names. Constructor for Java does not complain about a duplicate Component Name.

Modifying The Name

Each component has a name. The name can be edited in place in the The Project Window. Simply click on the text, and wait a moment. A frame appears around the item, and the text becomes selected inside the frame. You can then edit the text.

You can also edit the name in a Property Inspector window, as in Figure 3.4.

Figure 3.4 Property Inspector window



You cannot change the component type. However, you can modify the name.

Opening a Component for Editing

After you select a component, you can open an editor to modify the contents of the component. To open the editor associated with a component, you can:

- double-click the component
- select the component(s) and press the Return or Enter key

- select the component(s) and choose **Property Inspector** from the **Window** menu

You may open and edit several components of any type at the same time. If you open them simultaneously, Constructor staggers the editor windows on the monitor. Details of the various editor windows can be found in the appropriate chapters of this manual.

Deleting a Component

To delete a component or components, first select the component(s).

You may then delete the component(s) using one of three techniques. You may:

- choose **Clear Object** from the **Edit** menu
- press the Delete key
- drag the component(s) to the trash

You can Undo a delete until you perform another action.

4

Editing Layouts

This chapter discusses the tools you use and the tasks you perform to modify the contents of a layout using Constructor.

Editing Layouts Overview

We are at the core of what Constructor is all about. You want to quickly and easily specify the appearance of a layout and all of its contents. You can do so with Constructor.

This chapter discusses how to manage and modify the *contents* of a layout. For information on how to create, copy, modify, open, and delete layout components as a whole, see "Managing Components Overview."

The terminology in this chapter is important, and can be confusing. You will be using windows to create windows. You will have panels that contain panels. To minimize this confusion, here are a few terms and how they are used in this chapter.

- window—a display feature of Constructor! We will use the term "window" to refer only to Constructor's own interface.
- layout—the overall visual element you build with Constructor, stored in your Constructor project file. Typically, this is a window or panel that will appear in your own application or applet. We will call this a layout to distinguish it from Constructor's own windows.
- component—individual items inside a layout. These individual items are all instances of classes derived from the Component class. This chapter uses the term "component" to refer to layouts and components generically. They are, in fact, all derived from Component.

With the definition of terms out of the way, we can turn to the real subject at hand. Conceptually, this chapter consists of two sections:

what you see and what you do. The "what you see" section explains the Constructor windows involved in editing a layout. The "what you do" section discusses the tasks you perform. The topics are:

- Windows for Layouts—the windows you use to display and set layout-related information
- Building a Visual Interface—the tasks you perform to build a layout and its contents

Windows for Layouts

This section discusses the four Constructor windows involved in creating a layout and its contents. The principal features and characteristics of each window are discussed in detail. This section covers what you see in Constructor. The topics include:

- Catalog window—the window that contains the components you can add to any layout in any project
- Layout Window—the window that displays your layout's visual appearance
- Hierarchy Window—the window that displays the containment hierarchy for the components in your layout
- Layout and Hierarchy Compared—differences and similarities between these two windows
- Property Inspector Window—the window you use to set properties for a specific component

In addition, you examine component properties in the Property Inspector Window. There is one Catalog window and one Property Inspector Window for Constructor shared by all layouts. However, each layout has its own Layout Window and Hierarchy Window.

Catalog window

The *Catalog window* lists every class related to visual objects—that is, objects you can display. This window contains standard Java AWT component classes. It is the catalog you use as a source for new components when building a layout. See "Adding Components to a Layout."

To open this window, choose the **Catalog** item in the **Window** menu. This window shows the class name for each component.

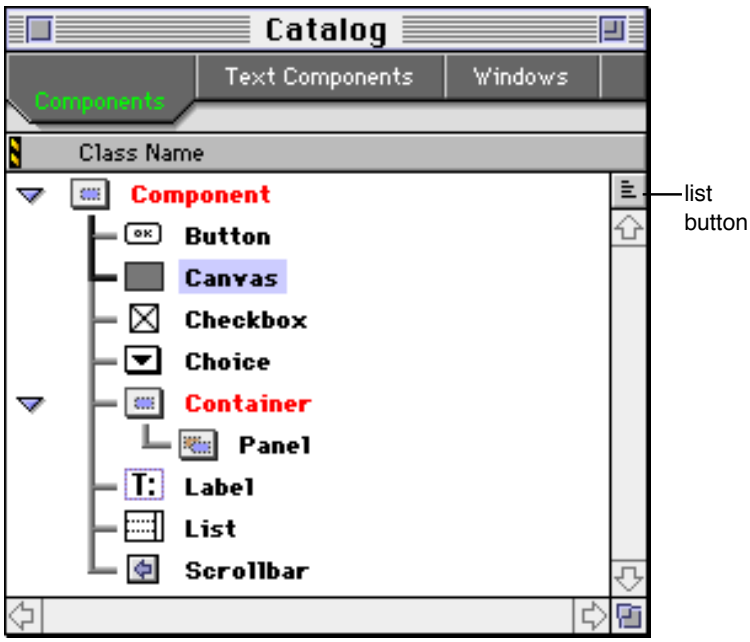
The components are organized into groups defined by an AWT base class. The groups are:

- Components
- Text Components
- Windows

To see the contents of a particular group of components, click the corresponding tab in the scrolling bar across the top of the window.

You can view the components by class hierarchy, as shown in Figure 4.1, or sort the panes by name. Click the List Button at the top of the vertical scroll bar to switch between hierarchy and name order.

Figure 4.1 The Java AWT Catalog window



Layout Window

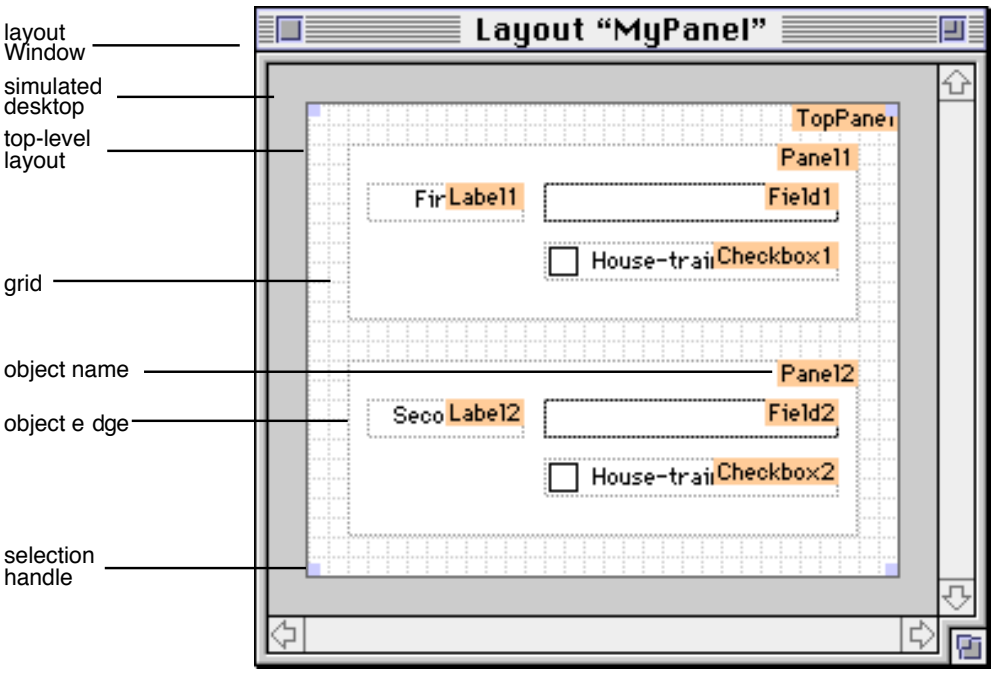
The layout window is where you do most of your visual work in Constructor. This is the window in which you arrange your components for functional and aesthetic appearance.

When you open a layout, its layout window appears. You may open the layout window in three ways:

- double click the layout in the project window
- select the layout and press the Return key
- select the layout and choose **Edit Layout** from the **Edit** menu

Figure 4.2 shows a layout window. The layout appears on a simulated desktop inside the layout window.

Figure 4.2 The layout window



The title bar of the layout window shows the class of the layout and its name. In this case it is a Panel named "MyPanel."

Within the layout window there is a layout. In this case it's an Panel. If you have a layout with a title bar (such as an Frame), you can see the title in the layout view. You can change the size of a top-level layout by dragging a selection handle.

The nature of the top-level layout depends on the type you select when you create it. Typically the top-level layout is an Panel, Frame, or Dialog. See "Making a New Layout."

TIP: You can't change the top-level layout type after you have created it. If you want to change the top-level layout, make a new layout with the correct type. Then copy the contents of the original layout to the new layout.

As you add components to the layout, you can rearrange them in the layout window, modify them, resize them, and so forth. See "Building a Visual Interface."

Hierarchy Window

The hierarchy window is an alternative but critical method of visualizing a layout and its contents. Components in a layout are arranged both spatially and hierarchically. A component that is visually inside another component may or may not be contained hierarchically within that component.

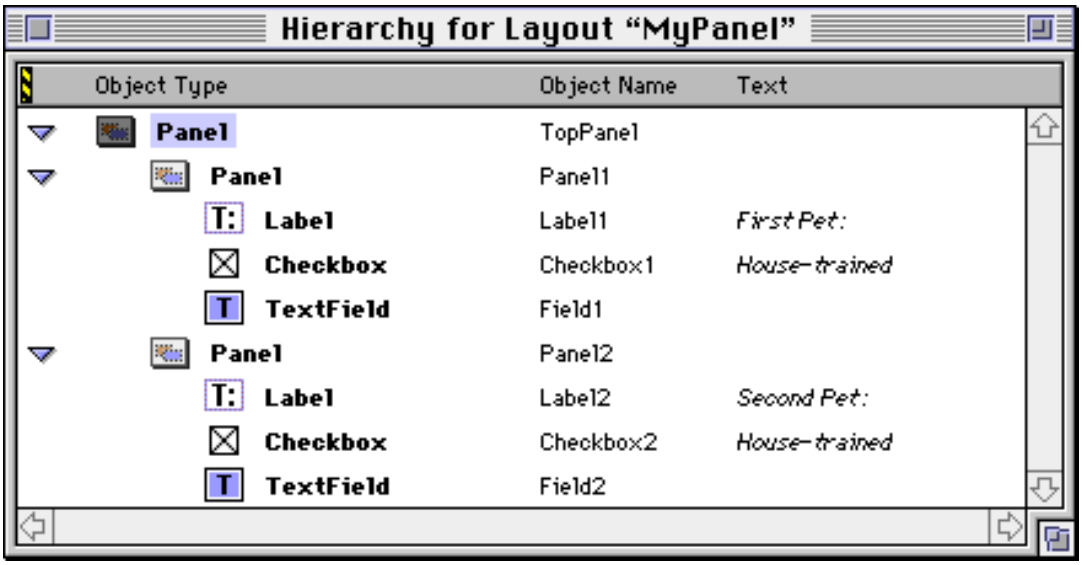
The containment hierarchy is an useful aspect of Java AWT programming. You can group together related components within a Panel, for example.

The containment relationships among components are stored with the layout, along with all the other layout-related information. Constructor use this information to set up the relationships when you reanimate the layout.

The hierarchy window displays the containment hierarchy for a layout. To display a layout's hierarchy window, choose the **Show Hierarchy** item (Command-H) from the **Layout** menu, when that layout window is active. The layout window must be open to view the hierarchy window.

Figure 4.3 shows a hierarchy window.

Figure 4.3 The hierarchy window



The top-level layout is always at the top of the hierarchy window. Triangles beside each layout let you show or hide their components.

NOTE: The hierarchy window also shows you the order in which components will be drawn. Your application will draw from the top of the hierarchy down. Components further down the list may hide those higher up if they overlap and draw over the same area.

You use the hierarchy window to rearrange the hierarchical order of the components in a layout. See "Arranging Components Hierarchically."

Layout and Hierarchy Compared

Constructor maintains a single JPob resource for each layout. The JPob contains all the data on all the components. The layout and hierarchy windows are simply different views on the same data set.

The layout window specializes in viewing the spatial relationships among the components in a JPob resource. You can see what they look like relative to each other, move them around, and so forth. You use the layout view to arrange the visual appearance of a JPob.

The hierarchy window specializes in viewing the containment relationships among the components in the JPob resource. You can see which components actually contain other components.

Either window provides access to the data in the JPob resource. Each looks at the data a little differently. However, you can use either view—layout or hierarchy—as a source or destination when copying components between projects. For example, you can copy a component from a layout window in one project, and paste it into the hierarchy window in another project.

Except that each window has a special way of viewing the JPob data, the layout and hierarchy windows are interchangeable. If you cut an item in the hierarchy window, it disappears from the layout. If you copy an item in the layout, it appears in the hierarchy. Constructor keeps both windows updated and consistent at all times.

Property Inspector Window

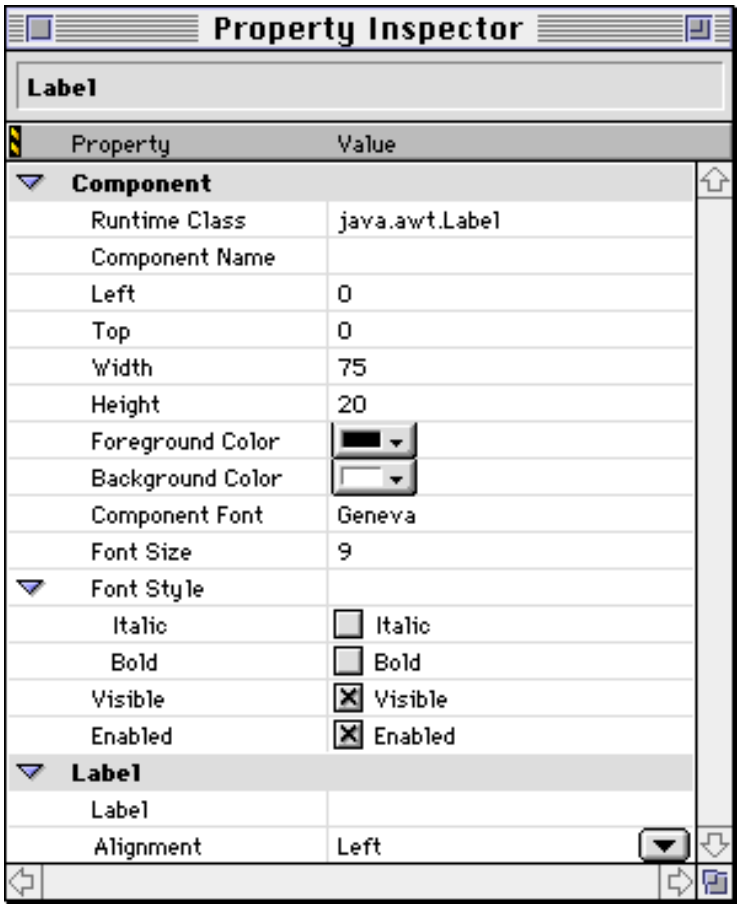
The property inspector window allows you to see and modify an individual component's data, such as location, size, content, and so forth. You open a property inspector using one of three methods:

- double-click the component
- select the component and press the Return key

- select the component and choose **Property Inspector** (Command-I) from the **Window** menu

Each component class has a different property inspector window, because each component has different properties. Figure 4.4 shows the property inspector for the Caption class.

Figure 4.4 The Label property inspector



The property inspector for most components contains these same fields. See "Setting Component Properties." "Component Properties Overview" discusses each property inspector and the data you can modify.

Building a Visual Interface

This section discusses the individual tasks you must perform when editing a layout. You must add components to a layout, arrange them visually and hierarchically, create certain groups, and set properties for the components. For information on how to create a new layout, see "Making a New Layout."

These tasks are not sequential. You perform these tasks concurrently, and in virtually any order that fits your personal style. However, at one time or another you will perform each task while creating a layout.

The topics in this section are:

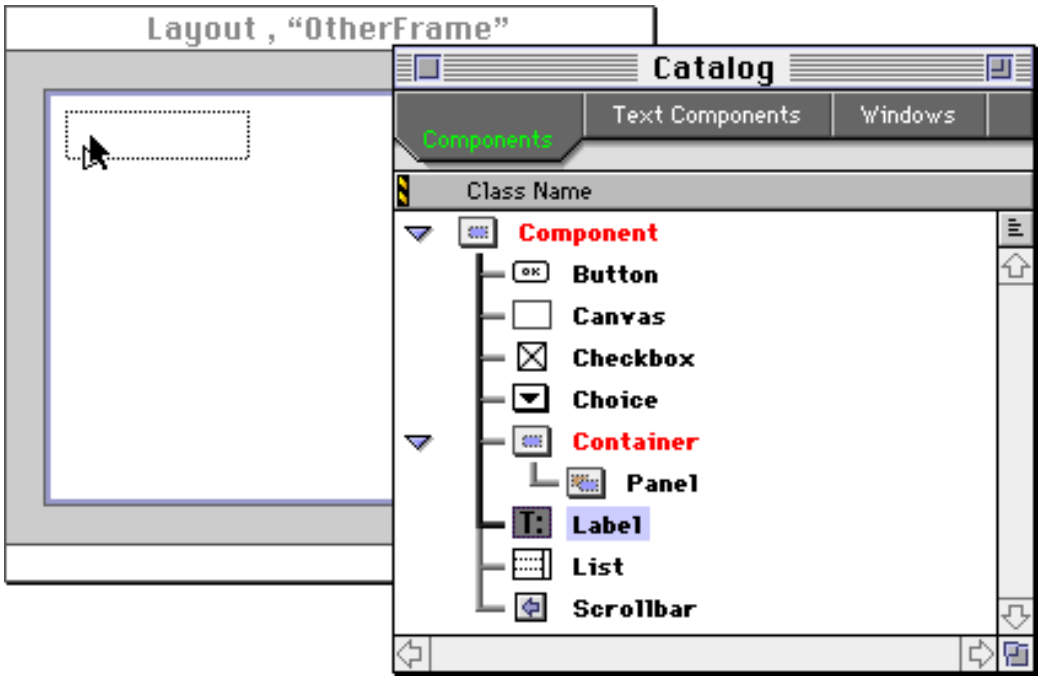
- Adding Components to a Layout—putting a new component into a layout
- Displaying Component Features—how to display component edges, invisible components, component IDs, and pictures inside components
- Arranging Components Visually—organizing components for an aesthetic display
- Arranging Components Hierarchically—organizing components in a containment hierarchy
- Setting Component Properties—specifying the contents and appearance of an individual component
- Removing Components—taking a component out of a layout
- Creating Customized Components—replacing Java AWT's components with your own customized versions

Adding Components to a Layout

To add a component to a layout, you copy a component from some other source and move it into the layout. The most common method is to drag a component from the display classes window into a layout. Figure 4.5 illustrates the process of adding an TextField to a layout.

You can drop components only into a layout. Constructor does not let you drop them on the simulated desktop around the layout.

Figure 4.5 Adding a component to a layout



A new component created by dragging from the display classes window always starts out with a default size and default attributes. For example, components designed for text are usually empty—that is, they have no text in them. You modify the component after adding it to the layout. See "Setting Component Properties."

NOTE: You cannot copy from the display classes window and paste into a layout. You must drag. There is no need to hold the option key down while dragging because you are copying from one container (the component catalog) into another (the layout window containing the layout you are building).

You can copy and paste components within a layout or between layouts. You use the standard techniques for copy and paste. See "Moving, Copying, and Deleting."

Displaying Component Features

When working in the layout window, you can control the display of certain component characteristics. Use menu commands for:

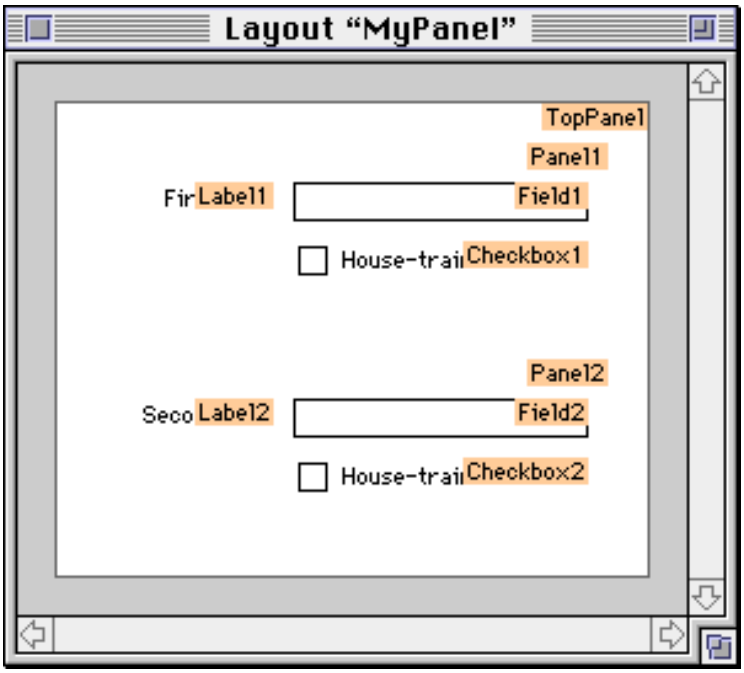
- Displaying component name—toggle the display of component ID numbers
- Displaying component edges—toggle the display of component boundaries
- Displaying invisible components—toggle the display of components that are invisible at runtime

Displaying component name

Each component has a name. The Reanimator class uses the name to find the component in memory. You set the component name in the property inspector for each component.

To display names, choose the **Show Object IDs** command in the **Layout** menu. The ID of each component appears. When the IDs are being displayed, the menu item text changes to **Hide Object IDs** . Choose **Hide Object IDs** to conceal the names. Figure 4.6 shows component names on display.

Figure 4.6 Component names on display



See "Using the Layout in your Java Program" for information on how to use component names.

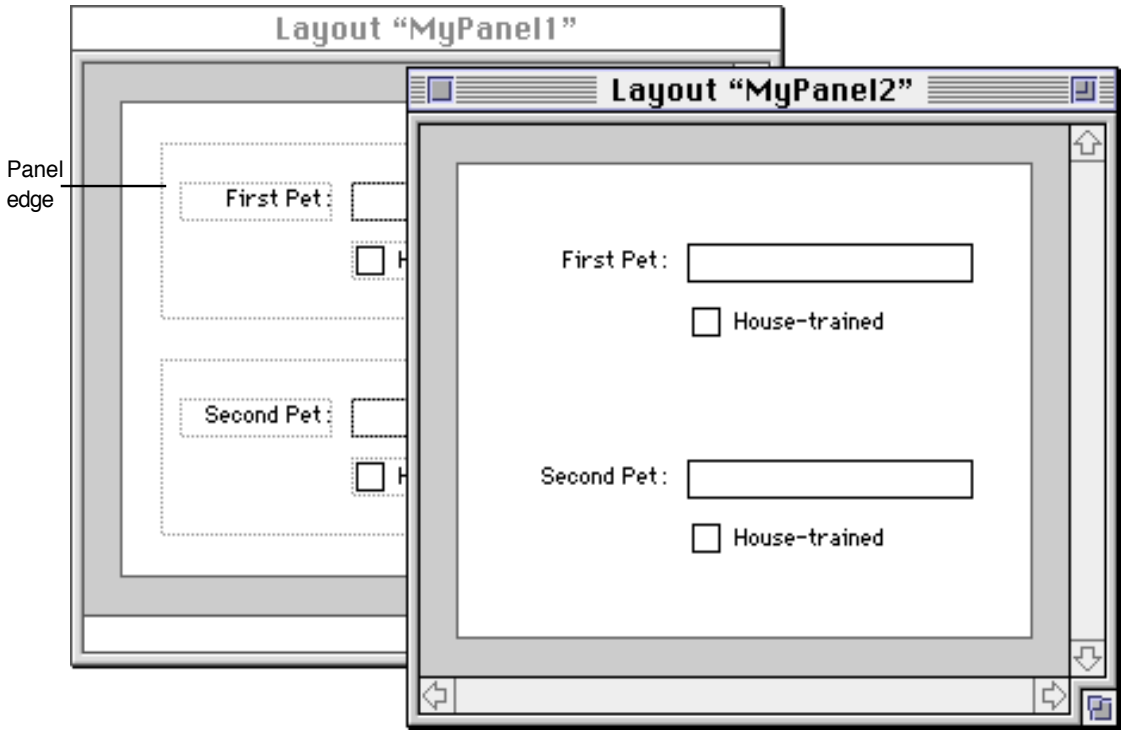
Displaying component edges

Every component has a boundary or edge. However, some component types have no actual visual representation. For example, an Panel can simply be a container for other components. If component edges are hidden, you cannot see a Panel object in the layout window. If you show component edges, you can see the dotted outline of the Panel component.

To display component edges, choose the **Show Object Edges** command in the **Layout** menu. The bounds of each component appear as a dotted outline. When edges are being displayed, the menu item text changes to **Hide Object Edges** . Choose **Hide Object Edges** to conceal the edges.

Figure 4.7 shows two layouts, each with two Panel objects. The component name is on display. In the window on the left, the component edges are visible. In the window on the right, the component edges are hidden, so the component bounds are invisible. If you hide both component names and edges, the Panel objects would be completely hidden. You could still select them, but you couldn't see them except for the selection handles while they are selected.

Figure 4.7 Object edges



Displaying invisible components

Each component has a *Visible* check box in the property inspector. You use this component attribute to determine whether a component is visible or not at runtime. However, Constructor reads the same attribute and allows you to show a component even if it is intended to be invisible at runtime.

To show non-visible components, choose the **Show Invisible Objects** command in the **Layout** menu. Choose **Hide Invisible Objects** to conceal the non-visible components once again.

Arranging Components Visually

This section discusses the various techniques you can use to arrange your components in a visually pleasing and aesthetic pattern within a layout. Constructor has features for:

- Selecting an obscured component—select a component behind another
- Moving a component—relocate components by dragging
- Nudging a component—move a component pixel-by-pixel
- Using a grid—display and/or snap to a definable grid
- Aligning components—align multiple components in various ways
- Distributing components—spread components evenly
- Resizing a component—change a component's size

Most operations require that you first select the component on components on which you wish to work, and then do something to them. You select components in the usual ways: click, Shift-click to extend a selection, or click and drag using the marquee. You can begin a marquee selection inside an exiting component if you hold down the Control key when you start the drag. See "Selection."

Selecting an obscured component

You select a component using standard selection techniques. See "Selection." However, you may have components stacked on top of each other such that one component completely covers (and thus obscures) a component behind it.

If you have one component on top of another, you can select the item *behind* the currently selected item by holding down the Command key while clicking. To select the next item in the stack, Command-click again. This cycles through the stacked components, including the top-level layout, without affecting any other

components. Continue Command-clicking until you have selected the desired component.

NOTE: If you have Command-clicked to select an obscured component, you must hold down the Option key before dragging, resizing, or double-clicking that component to prevent Constructor from changing the selection.

Moving a component

To move a component, drag the selection with the mouse and drop it at the desired location. You can select and drag multiple components as well. You may also open the property inspector for a component to change the component's location.

Hold down Shift and/or Command keys while dragging to constrain the movement of the component.

To constrain the movement to the same horizontal or vertical line, press the Shift key *after* you start dragging. This allows you to move a selection in a straight line. If you let go of the Shift key, the constraint is released and you can move the selection freely.

NOTE: Holding down the Shift key while selecting allows you to extend the selection.

To constrain the movement to a grid, you choose the **Snap to Grid** command in the **Layout** menu. To reverse the snap setting temporarily, press the Command key *after* you start dragging. If you let go of the Command key, the snap setting is determined from the menu.

NOTE: Holding down the Command key while selecting allows you to select a component behind the front component.

There is one component you cannot move. You cannot move a top-level layout. The top-level layout stays anchored in the layout win-

dow. You can reposition the location at which your top-level layout will appear in the desktop when you run your application. You do that by setting the location in the property inspector window for the top-level layout. See the *Java API Documentation* for details on positioning windows.

Nudging a component

Sometimes you want to move a component by one or two pixels. It can be difficult to be this precise dragging with the mouse.

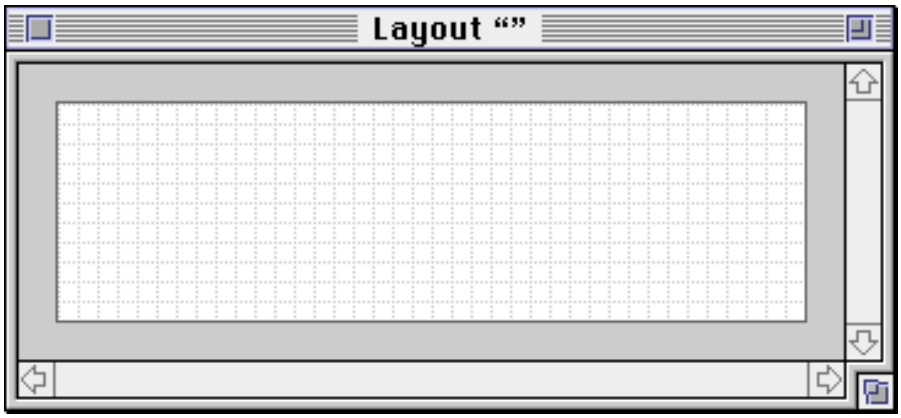
To nudge a component, first select the component. Then use the cursor control keys (the arrow keys) to move the component in the desired direction: up, down, left, or right. Each time you type an arrow key, the component moves by one pixel in the corresponding direction.

Using a grid

Constructor has a grid you can use to arrange components. You can display the grid, snap to the grid, and modify the grid spacing.

To see the grid, choose **Show Grid** from the **Layout** menu. To conceal the grid, choose **Hide Grid** . Figure 4.8 shows a 10-pixel square grid in an empty layout.

Figure 4.8 The Constructor grid on display

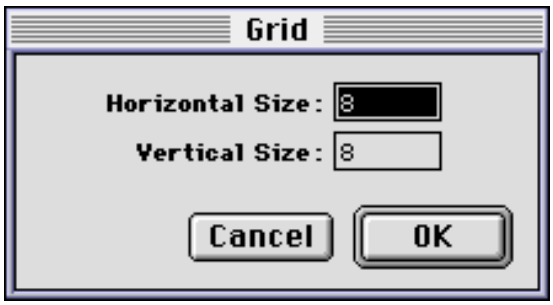


If you want all components to align along grid lines, choose the **Snap To Grid** command from the **Layout** menu. To allow free movement, choose **Don't Snap To Grid** . These commands have no effect on existing components. The snap to grid comes into play when you drag components. If the snap feature is on, the drag outline snaps to the grid while you drag an item. The snap feature works whether or not the grid is visible.

TIP: You can reverse the snap to grid setting when dragging components. Hold down the Command key after you start dragging.

You can set the grid spacing (in pixels). Choose the **Edit Grid…** item in the **Layout** menu. The Grid dialog appears as shown in Figure 4.9. Enter the grid spacing you want. You can have different values for horizontal and vertical spacing.

Figure 4.9 The Grid dialog



Aligning components

You can align selected components in the layout very easily. Constructor has three mechanisms for aligning components: menu commands, a palette, and a dialog. All three mechanisms have similar functionality for aligning components.

First, select two or more components you wish to align. Then issue the command for how you want to align the components. You may:

- choose an alignment command from the **Arrange** menu

- click a button in the alignment palette
- open a dialog by choosing **Arrange Objects…** (Command-L) from the **Arrange** menu

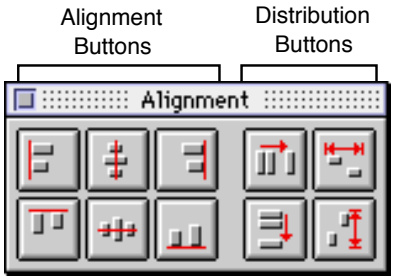
The alignment options, corresponding icons, and results are listed in Table 4.1. These correspond to menu commands in the **Arrange** menu, and buttons in the palette and the Arrange Objects dialog.

Table 4.1 Constructor component alignment options

Option	Icon	Alignment Line
Align left edges		the left of the leftmost selected component
Align horizontal centers		halfway between the left of the leftmost selected component and the right of the rightmost selected component
Align right edges		the right of the rightmost selected component
Align top edges		the top of the topmost selected component
Align vertical centers		halfway between the top of the topmost selected component and the bottom of the bottommost selected component
Align bottom edges		the bottom of the bottommost selected component

To see the alignment palette, choose **Alignment Palette** from the **Window** menu. Figure 4.10 shows the alignment palette, with the six alignment buttons on the left side of the palette. The other buttons are for distributing components. See "Distributing components."

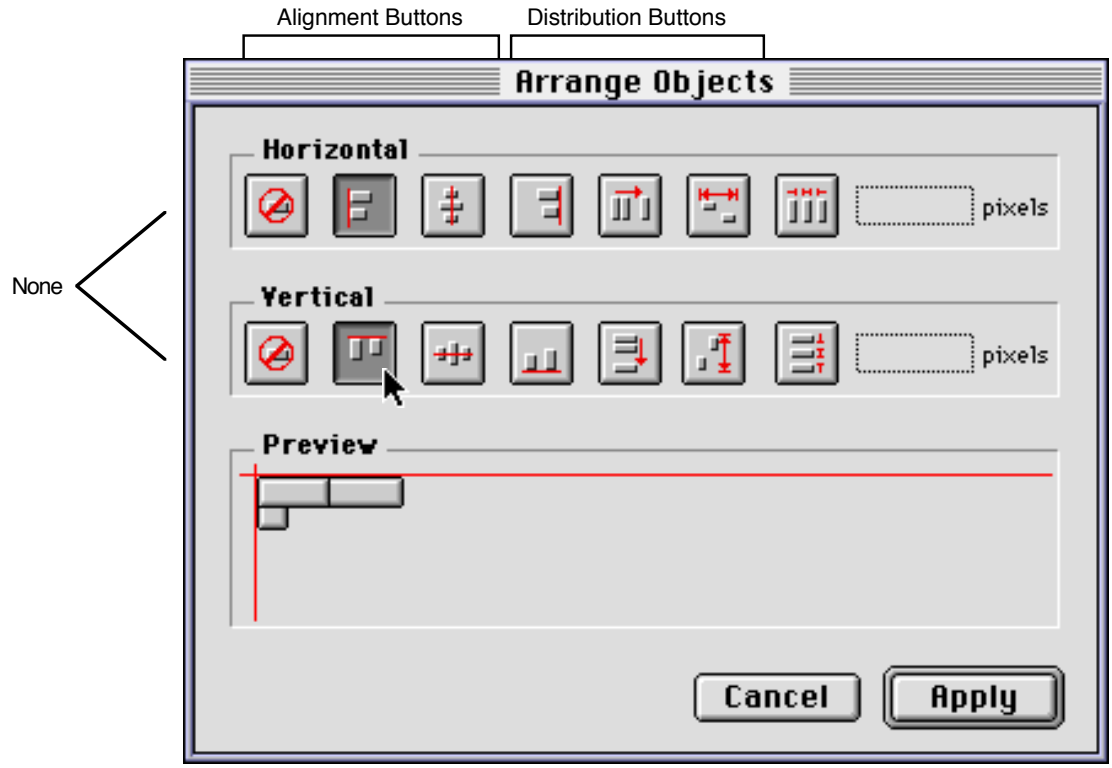
Figure 4.10 The Alignment palette



Simply click the appropriate button to obtain the desired alignment. The effect is immediate. Like most operations in Constructor, you can undo the action if you wish.

The third alternative for aligning components in Constructor is the Arrange Objects dialog. When you choose **Arrange Objects…** (Command-L) from the **Arrange** menu, Constructor displays the dialog shown in Figure 4.11. This dialog gives you a few alignment options not available with the alignment palette.

Figure 4.11 The Arrange Objects dialog



The Arrange Objects dialog lets you set both a horizontal and vertical alignment simultaneously, and lets you preview the effect of a command. The preview in the illustration shows objects aligned along top left edges.

Simply click the desired horizontal and vertical alignment. When you have set the desired alignment option, click the Apply button. The dialog disappears and your command is applied to the selected components. Constructor remembers the previous settings in the Arrange Objects dialog and restores them when you open the dialog again.

Distributing components

Distributing components means to spread them out evenly across a certain distance. Evenly means that after distribution, the gap between each pair of components is the same.

You can distribute selected components in the layout very easily. Constructor has three mechanisms for distributing components: menu commands, a palette, and a dialog. The dialog has additional features not available from menu commands or the alignment palette.

First, select two or more components you wish to distribute. Then issue the command for how you want to distribute the components. You may:

- choose a distribution command from the **Arrange** menu
- click a button in the alignment palette
- open a dialog by choosing **Arrange Objects…** (Command-L) from the **Arrange** menu

The distribution options, icons, and results are listed in Table 4.2. Some of these correspond to menu commands in the **Arrange** menu, and buttons in the alignment palette. The Arrange Objects dialog has all six options.

Table 4.2 Constructor component distribution commands

Option	Icon	Range of spread
Spread horizontally		between the left and right selected components
Spread horizontally in container		from the left edge to the right edge of the container that holds all the components
Spread horizontally by specified gap		from the left selected component, with the specified gap between components

Option	Icon	Range of spread
Spread vertically		between the top and bottom selected components
Spread vertically in container		from the top edge to the bottom edge of the container that holds all the components
Spread vertically by specified gap		from the top selected component, with the specified gap between components

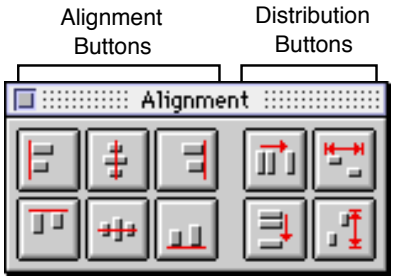
When distributed, components maintain their original left to right or top to bottom ordering, based on the left or top edges respectively.

You cannot specify a gap using the alignment palette or menu commands. You can specify a gap in the Arrange Objects dialog.

When distributing in a container, you must pay attention to the containment hierarchy. If all components are in the same container (such as a Panel, for example) then it is that container's bounds that serve as the range within which the components are distributed. If components are in multiple containers, the bounds of the container that holds all the selected components controls distribution.

Figure 4.12 shows the alignment palette, with four distribution-related buttons on the right side of the palette. The other buttons are for aligning components. See "Aligning components."

Figure 4.12 The Alignment palette

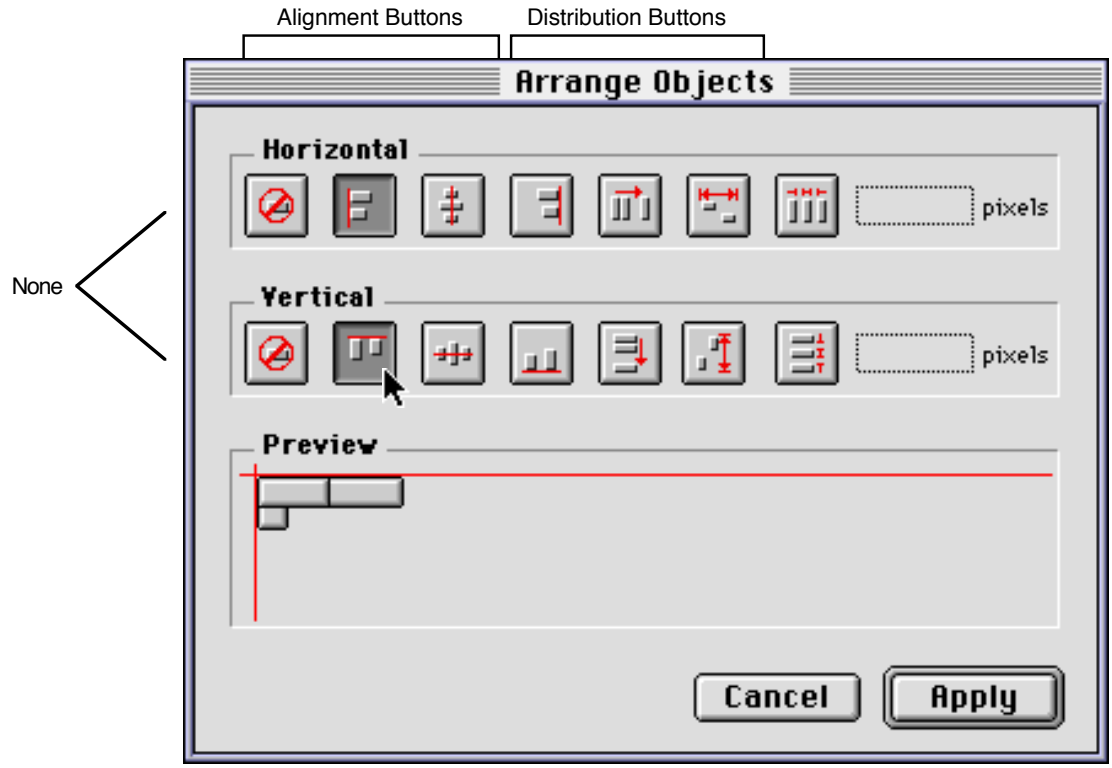


Simply click the appropriate button to obtain the desired distribution. The effect is immediate. Like most operations in Constructor, you can undo the action if you wish.

The third alternative for distributing components in Constructor is the Arrange Objects dialog. When you choose **Arrange Objects…** (Command-L) from the **Arrange** menu, Constructor displays the dialog shown in Figure 4.13. This dialog gives you a few distribution options not available with the alignment palette.

The Arrange Objects dialog lets you set both a horizontal and vertical distribution simultaneously, and lets you preview the effect of a command. The preview in the illustration shows objects distributed with a 10-pixel gap both horizontally and vertically.

Figure 4.13 The Arrange Objects dialog



You may also provide a set gap or space to use between components. To specify a gap, click the appropriate button. When you do, the *pixels* field associated with that button becomes enabled so you can enter the desired spacing.

When you have set the desired distribution options, click the Apply button. The dialog disappears and your command is applied to the selected components. Constructor remembers the previous settings in the Arrange Objects dialog and restores them when the dialog opens.

Resizing a component

To resize a component or components, click on one of the selection handles and drag it in any direction. You may also open the proper-

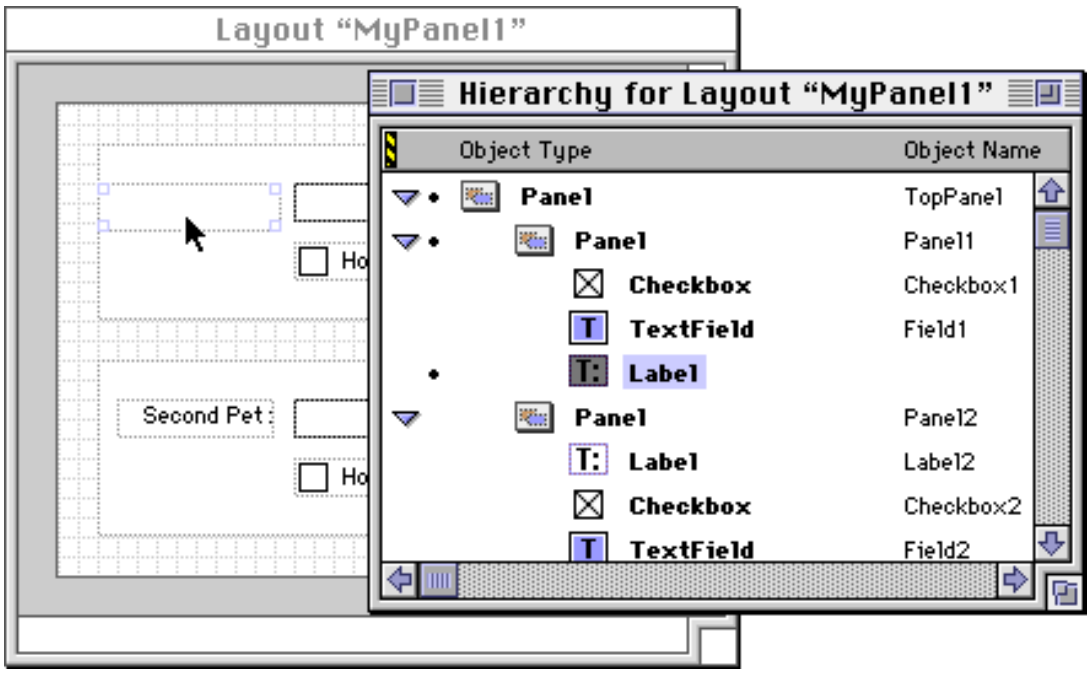
ty inspector window for a component and change the width and height. You can change the size of a top-level layout just like any other component.

Arranging Components Hierarchically

When you add a component to a layout, if the drop point is inside a containing component (like a Panel), then Constructor automatically places the new component inside the container, not only visually but hierarchically as well.

Figure 4.14 shows the result of dropping a Label component inside an existing Panel object. Notice in the hierarchy window how the Label component appears hierarchically inside the Panel.

Figure 4.14 Dropping a component in a container



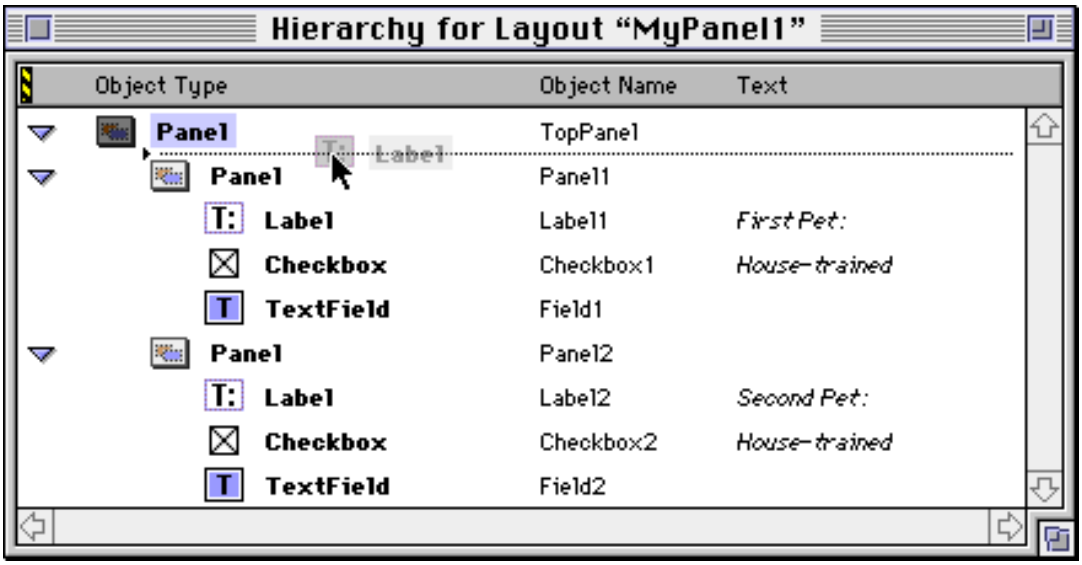
Notice that in the layout window you cannot tell whether the Label is actually contained in the Panel. They could simply overlap. The

hierarchy window shows you the containment relationship. There is no way you can tell the hierarchical relationship of these two components by looking at the layout window.

On the other hand, the hierarchy window is designed so that you can easily see and modify the component hierarchy.

To change a component's position in the containment hierarchy, select the component in the hierarchy window, and drag it to a new position anywhere in the hierarchy. While you drag, a small focus bar and arrow indicate the current destination, as shown in Figure 4.15. The focus bar flashes to enhance visibility.

Figure 4.15 Changing hierarchy order



In this illustration, the user is changing the Label object's container from one Panel to another. This does *not* change the Label object's spatial position in the layout.

You can reorganize multiple components simultaneously by selecting and dragging them as a unit.

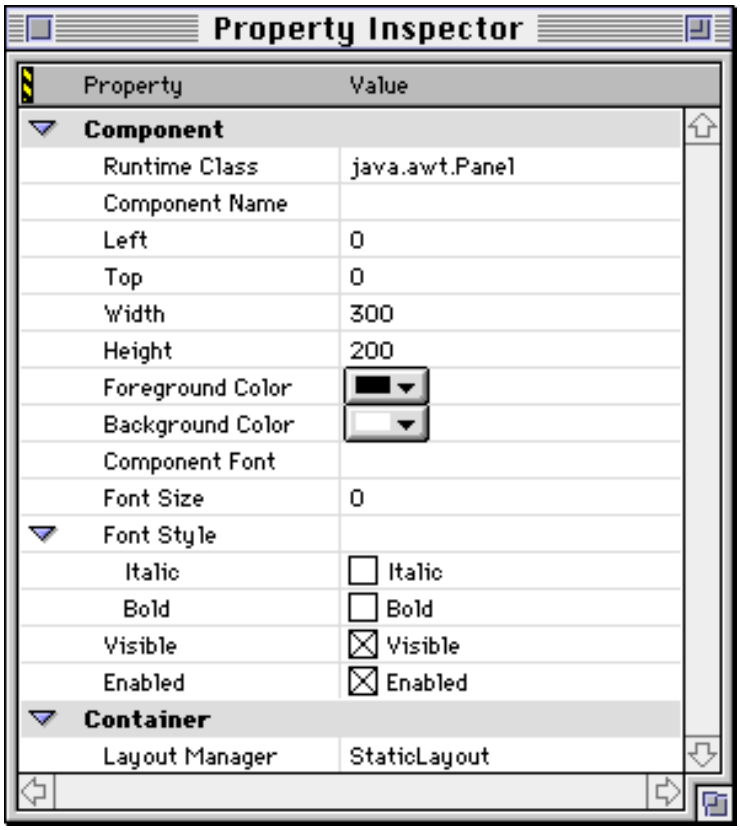
NOTE: Dragging a container operates on the container and subcomponents as a unit, rather than just the container. Constructor outlines the subcomponents as well as the container, so you can tell what you are moving.

Setting Component Properties

Each individual component has a variety of properties. For example, a component has a size and location. Different types of components have different properties. However, the process of setting component properties is identical for all components.

Open the property inspector window. Then modify the settings or fields in the property inspector.

Figure 4.16 The Panel property inspector



Each component property inspector presents all the customizable properties for that particular kind of component. See "Component Properties Overview" for a discussion of the many different components. For information on the effect or purpose of the various settings, see the *Java API Documentation.*

Removing Components

To remove a component from a layout, first select the component or components you want to remove. Then you may:

- press the Delete key
- drag the component to the trash

- choose the **Clear** command from the **Edit** menu
- choose the **Cut** command from the **Edit** menu

Items dragged to the trash are deleted immediately. They cannot be recovered from the trash. However, a delete operation can be undone until you issue a new command.

Items that are cut from the layout are placed on the clipboard and remain there until a subsequent cut or copy operation.

Creating Customized Components

When you first build your interface in Constructor, you can specify which classes the Reanimator routines will use when it re-creates your interface.

NOTE: This feature is not quite like adding custom pane types to Constructor for PowerPlant. You cannot add new component types to Constructor for Java. You can only replace the components it handles with customized versions.

For each component in your Constructor interface, you can specify a runtime class, which is the name of the class to instantiate for that component. By default, the runtime class is the name of a Java AWT class. However, you can change it to be another class, usually, a subclass of the default runtime class with your own customizations. Note that the class must be have a public constructor which matches the public constructor of the default runtime class.

Specify the runtime class in the Property Inspector window for the component, as shown below. For more information on the Property Inspector, see "Component Properties Overview."

Figure 4.17 Specifying the runtime class

When Constructor displays your component, it checks for a class with the same name of the runtime class. If such a class exists, Constructor creates an instance of that class and use it in the interface. Otherwise, it uses the default runtime class. To make sure Constructor can find your runtime class, put it in the Classes sub-folder of the Java folder inside the System folder.

5

Editing Menu Bars and Menus

This chapter discusses Constructor's menu bar and menu editors.

Editing Menu Bars and Menus Overview

Menu bars and menus are closely intertwined, both in Constructor and in a real application. A menu bar is little more than a list of menus. Constructor follows the same pattern. The menu bar editor displays a collection of menus and allows you to modify them. The two editors—menu bar and menu—are virtually identical.

This chapter discusses how to manage and modify the *contents* of menus. For information on how to create, copy, modify, open, and delete menu bars and menus as a whole, see "Managing Components Overview."

The major sections in this chapter are:

- Windows for Menus—the Constructor interface for menu editing
- Editing a Menu Bar—how to edit a menu bar
- Editing Menu Items—how to create and edit menu items

Windows for Menus

There are four menu-related windows involved in editing menu bars and menus. Each type of component has an editor, and a property inspector window. The four windows are:

- Menu Bar Editor Window
- Menu Editor Window

- Menu Property Window
- Menu Item Property Window

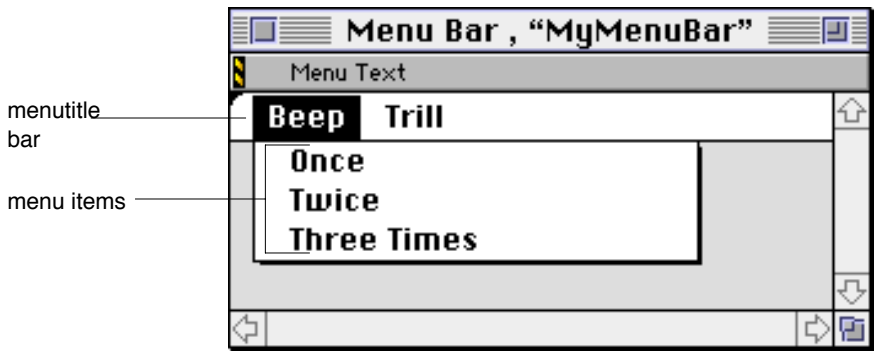
Menu Bar Editor Window

The menu bar editor displays a collection of menus contained in the menu bar. To open the menu bar editor window, you can:

- double click a menu bar in the project window
- select the menu bar and press the Return key
- select the menu bar and choose **Edit Resource** from the **Edit** menu

When you do, the window shown in Figure 5.1 appears. This illustration shows a typical menu bar.

Figure 5.1 The menu bar editor



The grey bar across the top of the window contains the column heads for the data displayed in the window. You see menu text, the and the menu command number for each menu item.

Like the layout window, the menu bar editor displays a simulated desktop. The white bar that contains menu titles is analogous to an actual menu bar on the Mac OS desktop, right down to the curved corner at the top left of the title bar.

In the grey area below the title bar, Constructor displays the actual menu items in each menu. The menu items appear just as they

would if you pulled the menu down in a running application. The Constructor menu bar editor is WYSIWYG—what you see is what you get—with two exceptions. Constructor does not display submenus interactively, although it does display the triangle indicator for a hierarchical menu item. Constructor also does not display an icon if the menu item has an icon attached.

A menu bar is really just a collection of menus. The Constructor menu bar editor allows you to see and edit not only which menus are in the menu bar, but the contents of each menu as well.

Menu Editor Window

The menu editor window is virtually identical to the menu bar editor. The only difference is that the menu editor displays a single menu. The menu bar editor can contain several menus.

To open a menu editor window, you:

- double click a menu in the project window
- select the menu and press the Return key
- select the menu and choose **Edit Resource** from the **Edit** menu

When you do, the window shown in Figure 5.2 appears.

Figure 5.2 Menu editor window

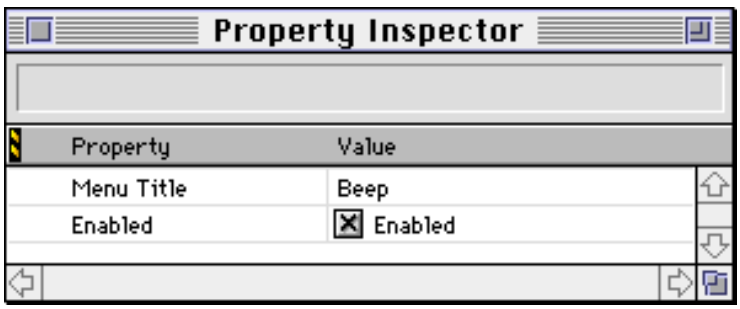
Menu Property Window

The menu property window displays data that applies to an entire menu. To open this window you:

- double click a menu title in the menu bar editor window
- select the menu title and press the Return key
- select the title and choose **Menu lnfo** from the **Edit** menu

When you do, the window shown in Figure 5.3 appears.

Figure 5.3 Menu property window



You can modify the menu title and choose whether the menu is enabled by default when it's created.

Menu Item Property Window

The menu item property window displays data that applies to a particular menu item. To open this window you:

- double click a menu item in either the menu bar editor or the menu editor
- select the menu item and press the Return key
- select the menu item and choose either **Edit Menu Item** or **Menu Item lnfo** from the **Edit** menu

When you do, the menu item property window shown in Figure 5.4 appears. You can modify many attributes of a menu item. Following

the illustration, Table 5.1 list each item in the window and the menu attribute it specifies.

Figure 5.4 Menu item property window

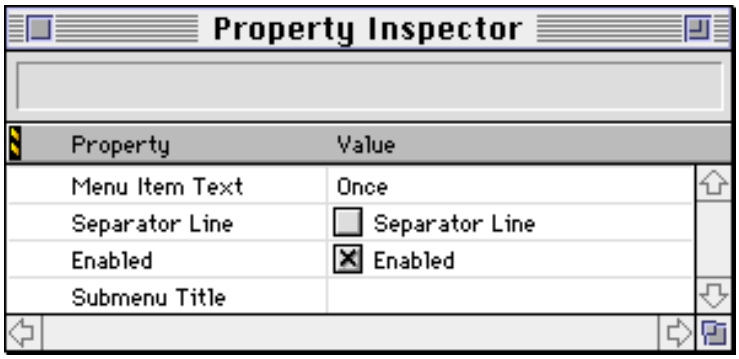


Table 5.1 Menu item properties

Item	Specifies
Menu Item Text	menu title
Separator Item	whether this item is a separator bar
Enabled	menu item is enabled by default
Submenu Title	menu title for a submenu if this is a hierarchical menu item

Editing a Menu Bar

This section discusses the tasks you perform to manage and edit a menu bar. All of these tasks are performed within the menu bar editor. The topics discusses in this section are:

- Making a New Menu—creating a new menu in the menu bar
- Adding an Existing Menu—adding an existing menu to the menu bar
- Modifying a Menu—changing a menu's title and other characteristics

- Changing Menu Order—changing the order of appearance of individual menus in the menu bar
- Removing a Menu—deleting a menu from the menu bar

For related topics, see:

- "Creating a Menu Bar"—how to create a menu bar
- "Menu Bar Editor Window"—how to open the menu bar editor
- "Editing Menu Items"—how to edit particular menu items

Making a New Menu

To make a new menu, choose the **New Menu** item in the **Edit** menu. When you do, a new menu title appears in the menu bar editor to the right of the currently selected menu title. If there is no selection, the new menu is added to the end of the menu bar.

If the new menu is one of the first three menus added to the menu bar, Constructor automatically assigns the correct menu title: Apple icon, File, or Edit. For subsequent menus, the initial menu title is "untitled."

When you create a new menu, the menu title is placed in editing mode so you can modify the text. To end editing mode, click in an empty place in the editor window, or press the Tab or Return keys.

Creating a new menu does not add items to the menu. For more information on this topic, see "Making a New Menu Item."

Adding an Existing Menu

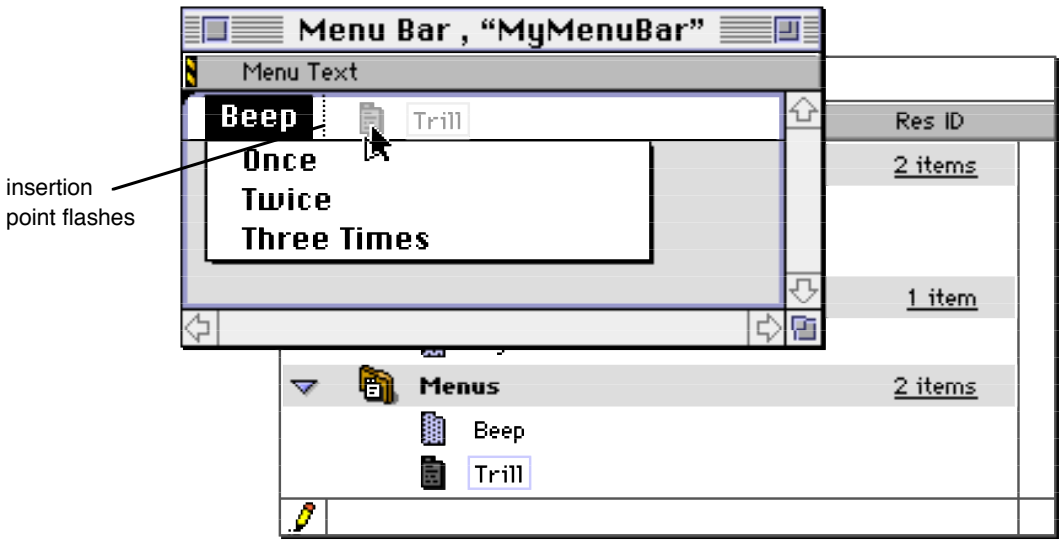
If you have an existing menu that is not already in the menu bar, you can add it. Drag the desired menu from the project window and drop it into the menu bar editor.

In the project window, select the menu(s) you want to add to the menu bar. It must not be in the menu bar already. You can tell which menu are not in the menu bar, because the icon to the left of the component is clear and not grey. In Figure 5.5, look at the differ-

ence in the icons for the **Edit** menu and the other two menus. The icons for menus that are already in the menu bar are grey because they are open.

Drag the selected menu(s) from the project window into the menu bar editor area, as shown in Figure 5.5. As you do, a small vertical line will flash in the menu bar title area indicating the insertion point for the menu you are adding. Drop the menu where you want it to appear in the menu bar.

Figure 5.5 Adding an existing menu



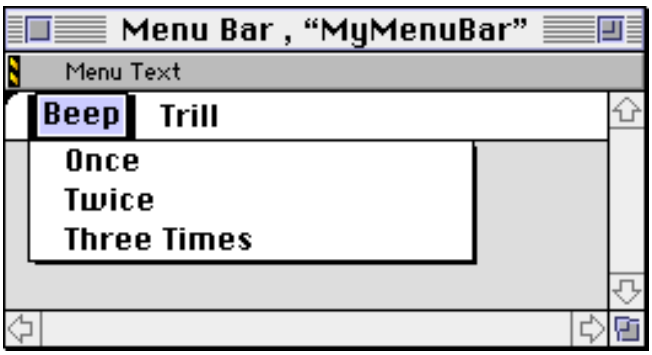
Notice that you don't have to drop the menu in the actual title area. You can drop it anywhere in the editor window. The insertion point will track your mouse movements horizontally to show you where the menu will appear in the menu bar.

TIP: If you select multiple menus, all must be absent from the menu bar or none will be added when you drop the menus.

Modifying a Menu

To change the menu title, you can put the text into edit mode. Then you can modify the text in place, as shown in Figure 5.6. See "Text Editing" for more information.

Figure 5.6 Changing a menu title

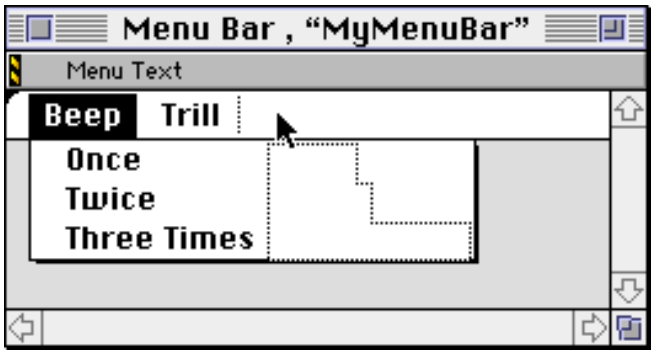


You can change the menu title text in the menu property window as well. Open the Menu Property Window by double-clicking the menu title.

Changing Menu Order

To change the order of menus in the menu bar, simply drag a menu to a new location. A flashing insertion point indicates where in the menu bar the menu will appear when you drop it. Figure 5.7 illustrates the process of moving the **Beep** menu to the right of the **Trill** menu.

Figure 5.7 Moving a menu



Removing a Menu

To remove a menu from the menu bar, first select the menu. Then you can:

- press the Delete key
- choose **Cut** or **Clear** from the **Edit** menu
- drag the menu to the trash

TIP: When you click the menu title to select it, it is likely to go into edit mode. Simply press the Return or Tab keys, or click in an empty part of the window to turn off edit mode.

Editing Menu Items

This section discusses the tasks you perform to manage and edit the items that make up a menu. For information on modifying the menu title, or information related to the menu as a whole, see "Editing a Menu Bar."

All of the tasks discussed in this section may be performed within either the menu bar editor or the menu editor. The functionality of the menu bar and menu editors is virtually identical. The menu bar editor is simply a means of accessing a series of menus in the same window. The menu editor displays one menu.

As a practical matter, the menu bar editor is a lot more useful, because you can work with several menus simultaneously. However, it does not display submenus. Use the menu editor for submenus and other menus not in the menu bar, such as pop-up menus.

The topics discussed in this section are:

- Selecting a Menu Item—how to select menu items and control edit mode
- Making a New Menu Item—adding a new menu item
- Making a Menu Separator Item—adding a separator
- Modifying a Menu Item—changing item text, and setting other properties for an individual menu item
- Moving a Menu Item—shuffling menu items
- Adding a Submenu—creating a hierarchical menu item
- Removing a Menu Item—deleting a menu item

For information on related topics, see also:

- "Creating a Menu"—how to create a menu
- "Menu Bar Editor Window"—how to open the menu bar editor
- "Menu Editor Window"—how to open the menu editor

Selecting a Menu Item

Working with menu items usually requires that you select one or more items. When you select an item, you may or may not want it to go into edit mode for in-place text editing.

There are four ways you can select menu items.

1. **Arrow keys** . Use the up or down arrow key to select the previous or next item in the menu. Press the shift key to extend the selection.
2. **Click text** . Click text in a menu item. If you move the cursor off the item or wait one second, the item will go into edit mode for in-place editing. You can click the item text, the Command-key character or the command number to put them into edit mode.

3. **Click the menu line** . Click the white space to the left or right of a menu item. This selects the item without going into edit mode. Press the shift key to extend the selection.
4. **Selection marquee** . Selects any intersecting item without going into edit mode.

You can also choose **Select All** from the **Edit** menu to select all the items in a menu.

TIP: Use the left or right arrow key to change menus. Although you can select multiple menu items, you cannot select multiple menus.

You can get into and out of edit mode at will.

Getting into edit mode Click on the text and move the mouse off the text (or wait a second). Alternatively, you can also select a single item and press the Tab key to enter edit mode.

Getting out of edit mode Press the Return key. The item remains selected but edit mode is turned off.

Making a New Menu Item

To add a new menu item to a menu, choose **New Menu Item** from the **Edit** menu. The new item is added after any existing selected menu item, or at the end of the menu if there is no selected item. New items are named "untitled item," and have a menu command number of zero. Make sure you modify the command number appropriately.

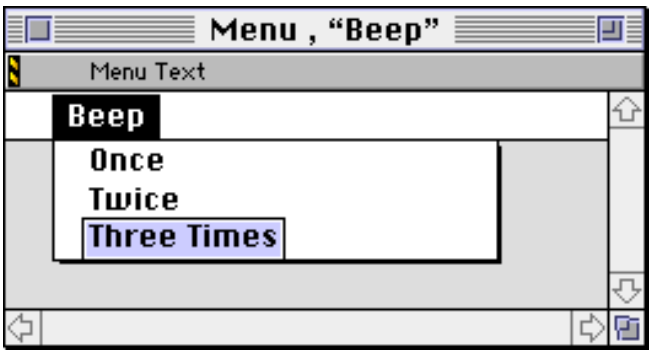
You can copy, paste, and duplicate selected menu items. A pasted item appears after any selected item, or at the end of the menu if there is no selected item. Duplicated items appear immediately after the original. If an item is in edit mode, editing operations affect the item text, not the item as a whole.

If you have two or more menu editor windows open simultaneously, you can drag a menu item from one window to another to create a copy.

Modifying a Menu Item

The editor window displays the menu item text and the command number. These items may all be edited in place. Select the item and put it into edit mode. Then edit the item. For information on selecting an item and entering edit mode, see "Selecting a Menu Item."

Figure 5.8 Editing a menu item in place



When editing a menu item in place, the Tab key advances you to the next part of the item. You can move from item text to command number with the Tab key.

Use the Menu Item Property Window to see and modify other item-related properties. You can open the item property window by double-clicking a menu item. See "Menu Item Property Window" for a detailed discussion.

Making a Menu Separator Item

You can make a separator item using any of these techniques:

- Choose **New Separator Item** in the **Edit** menu.
- Make a new menu item. When the item appears in edit mode, make the item text a single hyphen. Press Return when you have made this change, and the separator appears.
- Open the Menu Item Property Window for an item and turn on the *Separator Item* check box.

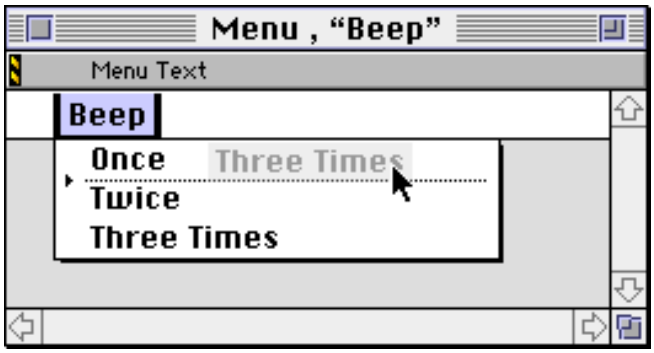
You cannot put a separator item into edit mode. To convert a separator back into a regular menu item, use the menu property window and turn off the *Separator Item* check box. You can then edit the menu text normally.

You can also copy and paste, duplicate, or copy a separator item by dragging.

Moving a Menu Item

To reposition a menu item in a menu, simply drag it to a new location in the menu. Figure 5.9 illustrates the process.

Figure 5.9 Moving a menu item



There is a small arrow and a flashing line that indicate the insertion point in the menu.

You can change the order of menu items using cut and paste. A pasted item is placed in the list after the currently selected item, or at the end of the menu if there is no selected item.

You can also drag a menu item from one menu to another in the menu bar editor. Drag the item to the menu bar. The menu bar editor dynamically switches menus as you drag the item across the menu bar. When you reach the proper destination menu, drag down the menu to drop the menu item in the correct location.

Adding a Submenu

To make a menu item a hierarchical menu item, you must attach a submenu to it. To do so, open the Menu Item Property Window. Then enter the name of a menu in the Submenu Name field.

Note that no hierarchical triangle mark appears to the right of the menu item in the menu editor window. However, the hierarchical menu will appear correctly in your finished application.

Removing a Menu Item

To remove one or more menu items from a menu, first select the item or items. Then you can:

- press the Delete key
- choose **Cut** or **Clear** from the **Edit** menu
- drag the items to the trash

The items disappear from the menu.

6

Component Properties

This chapter is a reference for the properties in each individual component as they appear in the Property Inspector Window.

Component Properties Overview

Every kind of component has stored data. Some are very simple, others are more complex. You modify and view this data in the property inspector for the individual component.

This chapter discusses how to work with a property inspector, and how the structure of the window reflects the nature of the component. The display of data in the Property Inspector Window is highly modular.

The rest of the chapter is a class-by-class reference to the individual property inspector for each component in AWT.

The topics in this chapter are:

- Using a Property Inspector Window—working with a property inspector
- Anatomy of a Property Inspector—the parts of a property inspector, and how each window is described in this chapter
- Component—the base properties for all components
- Layouts—properties for components that contain other components
- Controls—properties for control classes

In general, the information in this chapter does not tell you the purpose for each data item in a property inspector, or how to use that

item to best effect in a Java application. See the *Java API Documentation* for information on class data members, and how to use them in a Java application.

Using a Property Inspector Window

You open a component property inspector in Constructor for Java using one of three methods:

- double-click the component in the layout or hierarchy window
- select the component and press the Return key
- open the Property Inspector Window and select a pane

The property inspector displays various options that you can modify. Each property inspector is, essentially, a modeless dialog box of sorts. You use standard techniques to modify the contents. Use the mouse to change options and enter fields, or navigate through the edit fields with the Tab and Shift-Tab keys.

Changes in a property inspector take effect in the layout and hierarchy windows quickly. Check box and radio button changes show up immediately. Changes in the edit fields appear as soon as you leave the field by pressing Tab or Enter, clicking elsewhere on the screen, or closing the property inspector.

When editing, all normal editing functions apply, including Cut, Copy, Paste and Clear. You can navigate through the text with arrow keys. You can extend the text selection by pressing the Shift key as you click the mouse.

You can undo any change that you make by choosing **Undo** from the **Edit** menu (Command-Z). The menu item will be either **Undo Info Change** or **Redo Info Change** .

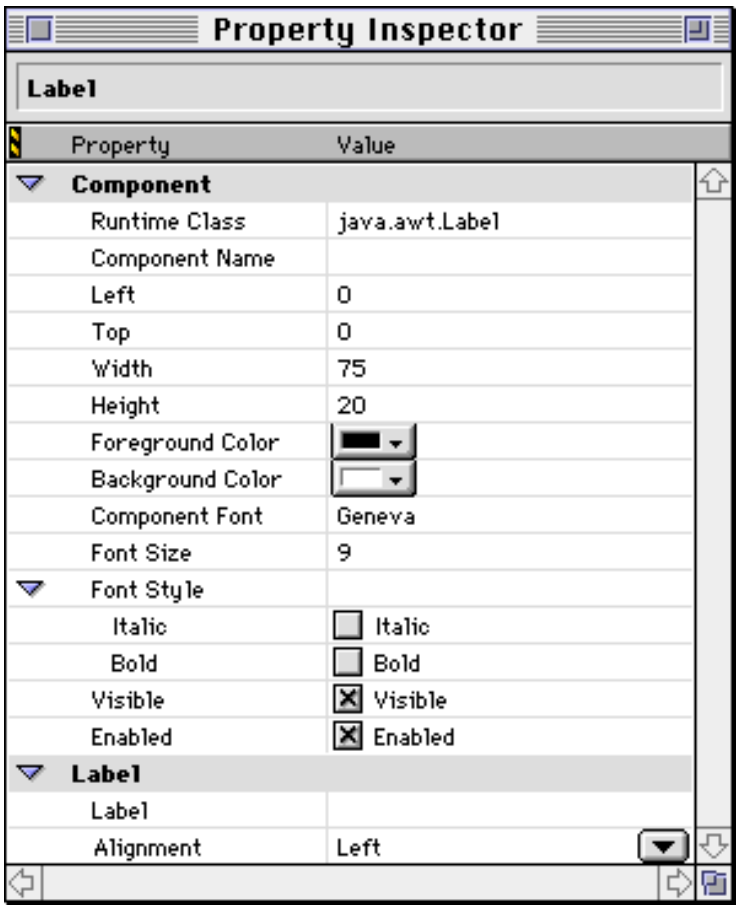
Anatomy of a Property Inspector

The property inspector looks like a spreadsheet, with property names in one column and property values in the next. The property

inspector for a particular type of component reflects its place in the AWT class hierarchy. Each of the class's superclasses has its own section, headed by a gray bar with the class's name. The superclasses are ordered starting with the base class to the actual component class itself. The class java.awt.Component is the base class for this entire hierarchy, and every type of component has its properties.

For example, Figure 6.1 shows the property inspector for an Label object. Notice that the top part of the window has Component properties, and the bottom part of the window contains the label-specific data.

Figure 6.1 Property inspector for a Label



Each property inspector is structured similarly. Properties from Component appear at the top. Properties specific to the component appear at the bottom.

Dots appear besides fields whose values you've changed since you last saved the file. If a class section contains any changed values, a dot appears besides the class header as well.

Triangles appear besides all class headers and certain properties that represent member classes (such as Font Style). These triangles let you hide or show that class's properties. If you click on a down arrow, it turns to the right and hides those properties. If you click on a right arrow, it turns down and displays the those properties.

In the topics in this chapter, information is presented consistently. The discussion of each class has these sections.

- **Description** —a brief description of the component
- **Inheritance** —the *visual* class hierarchy for this component, arranged with base class to the right (does not include non-visual classes)
- **Properties** —the data specific to this component
- **Special considerations** —extra information and tips regarding this component's properties

For each class the properties section has an illustration showing the data items specific to that class. Each data item is also explained in a table like this:

Table 6.1 Sample table of properties

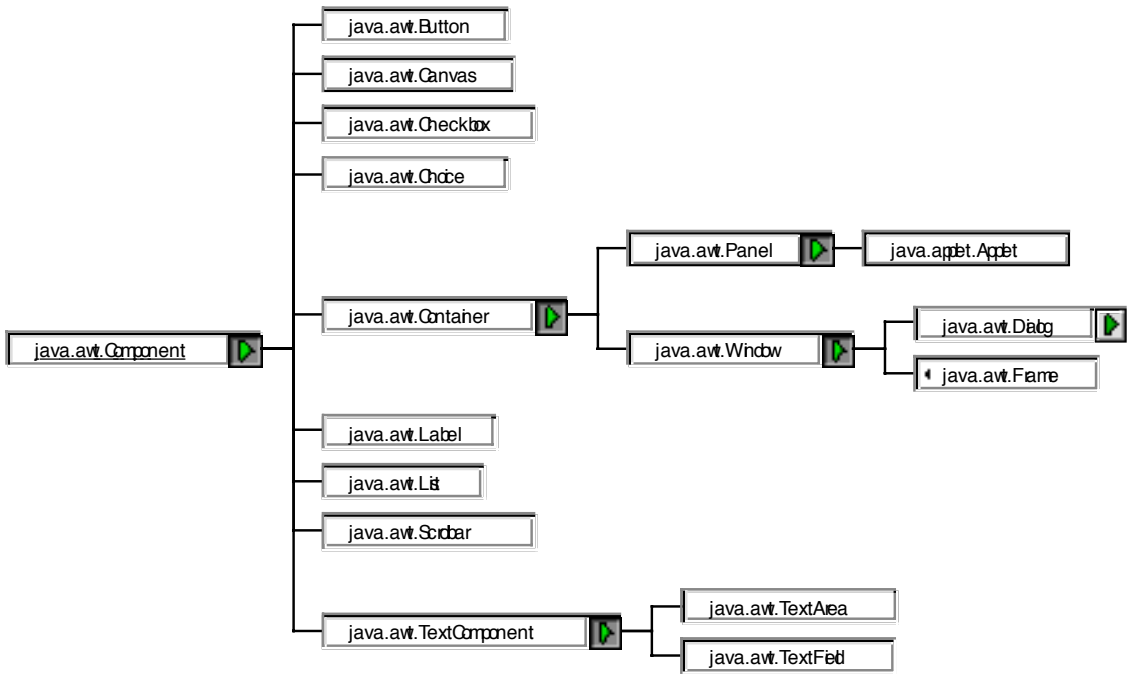
Item	Specifies
Top/Left	The top left corner of the component in its layout.

The actual property inspector for any component is typically built out of the properties from each ancestor class, and the specific properties for the component in question. To see all the properties for a

component, refer to the discussion for each ancestor class as well as the specific component of interest.

For those cases where the property inspector of a component is not the sum of all ancestor components, that fact is noted in the description of the properties.

Figure 6.2 Hierarchy for Component



Component

Description The base class for all graphic user interface objects in AWT, except for menus. You cannot add a pure Component to a JPob, but all other objects in a JPob inherit from Component.

Inheritance None. This is the base class for all graphic user interface objects in Constructor for Java.

Properties Figure 6.3 shows the property inspector for a Canvas, which doesn't contain any properties of its own. All property inspectors display this data at the top of the window.

Figure 6.3 Property inspector for Component

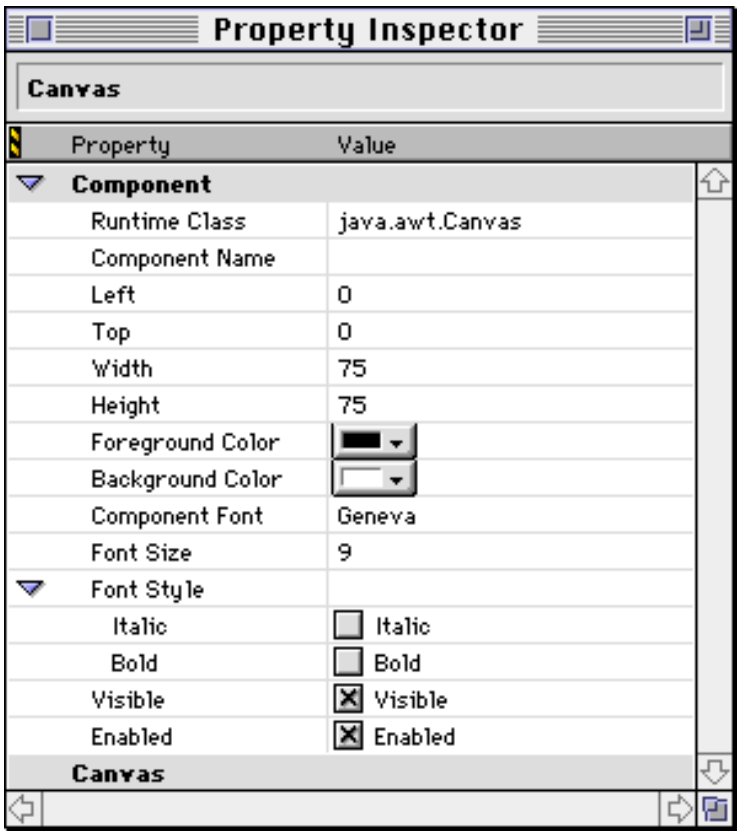


Table 6.2 Component properties

Item	Specifies
Runtime Class	The name of the class.
Component Name	A string that uniquely identifies the component in its layout.

Item	Specifies
Left/Top	The top left corner of the component in its layout.
Width	The width of the component.
Height	The height of the component.
Foreground Color	The color of elements in the component's foreground, such as text or graphics.
Background Color	The color of the component's background.
Font Style	The font style for text in the component.
Italic	Whether the text is italic.
Bold	Whether the text is bold.
Visible	Whether the component is visible at runtime.
Enabled	Whether this component receives events.

Special considerations The default value for the Runtime Class is the name of the AWT class for the component. You can change this to be the name of another class that implements your own customized version for that type of component. For more information see "Creating Customized Components."

The default value for Component Name is the empty string. Every component needs a name, so the Java code that recreates your layout can do its work. The name must be a valid Java identifier; that is, it can contain only alphanumeric characters and underscores, and it cannot contain spaces or punctuation marks.

Layouts

A layout is a Component that can contain other Components. When you first create a JPob, you must create a layout before you can add buttons, checkboxes, text fields, and other user interface elements.

The layout classes are

- Container
- Window
- Dialog
- Panel
- Frame

Container

Description A base class for all the layouts. You cannot add a Container to a JPob, but all other layouts are derived from it.

Inheritance Container -> Component

Properties Figure 6.4 and Table 6.3 show the properties specific for LFinder.

Figure 6.4 Properties for Container

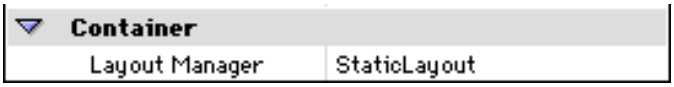


Table 6.3 Container properties

Item	Specifies
Layout Manager	The name of the layout manager for this layout.

Special considerations Generally, you won't change the Layout Manger. The default value, StaticLayout, is a class that makes sure that all the components in your final application or applet are arranged just as they are in Constructor for Java. If you change the Layout Manager, the components may be arranged differently.

Panel

Description A layout that is contained within other layouts. It does not create its own window.

In an applet, use a Panel as the base layout. In any program, you can use a Panel inside another layout to group together related components.

Inheritance Panel -> Container -> Component

Properties Same as Container.

Special Considerations None.

Window

Description A top-level window with no title, no border, and no menu bar. Although Constructor for Java lets you use a Window as a layout, you'll usually use one of its subclasses, instead: Frame or Dialog.

Inheritance Window -> Container -> Component

Properties Same as Container.

Special Considerations None.

Frame

Description A top-level window with a title, a border, and, optionally, a menu bar.

Inheritance Frame -> Window -> Container -> Component

Properties Figure 6.5 and Table 6.4 show the properties specific for Frame.

Figure 6.5 Properties for Frame

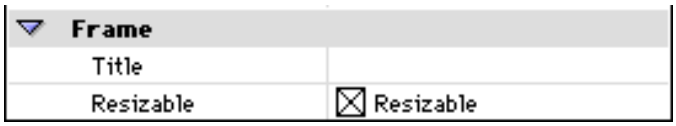


Table 6.4 Frame properties

Item	Specifies
Title	The title of the window. On the Mac OS, this appears in the window's title bar.
Resizable	Whether the window is resizable. On the Mac OS, a resizable window has a size box in the lower right corner.

Special considerations None.

Dialog

Description A dialog window. This class is intended for temporary windows, which present information to the user or let the user to specify options for the current operation.

Unlike all other layouts, a dialog window is invisible by default. You must use the `show()` method to display it.

Inheritance Dialog -> Panel -> Container -> Component

Figure 6.6 Properties for Dialog

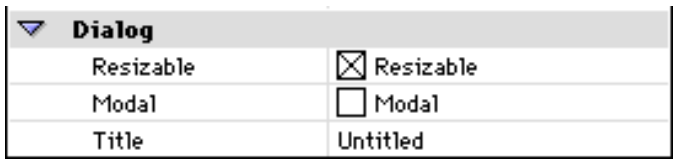


Table 6.5 Dialog properties

Item	Specifies
Resizable	Whether the window is resizable. On the Mac OS, a resizable window has a size box in the lower right corner.

Item	Specifies
Modal	Whether the window blocks all other windows from getting input.
Title	The title of the window. On the Mac OS, this appears in the window's title bar.

Special considerations None.

Controls

The control classes represent buttons, check boxes, scroll bars, and text. The control classes are

- Button
- Checkbox
- Label
- Scrollbar
- TextField
- Canvas
- Choice
- List
- TextComponent
- TextArea

Button

Description A push button with a label.

Inheritance Button->Component

Properties Figure 6.7 and Table 6.6 show the properties specific for Button.

Figure 6.7 Properties for Button

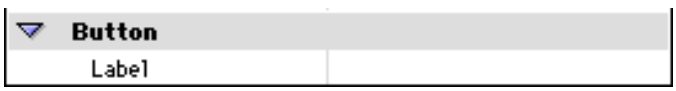


Table 6.6 Button properties

Item	Specifies
Label	Text in the button.

Special considerations None.

Canvas

Description A blank rectangular area of the screen that you can use for drawing or trapping input events.

Inheritance Canvas -> Component

Properties Same as Component

Special considerations None.

Checkbox

Description A check box that can be on or off. Clicking it toggles its value.

NOTE: This version of Constructor for Java does not support CheckboxGroup.

Inheritance Checkbox -> Component

Properties Figure 6.8 and Table 6.7 show the properties specific for Checkbox. The Checkbox property inspector is a combination of these properties and Component.

Figure 6.8 Properties for Checkbox

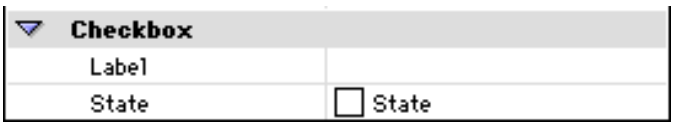


Table 6.7 Checkbox properties

Item	Specifies
Label	Text in the checkbox.
State	Initial value of the checkbox.

Special considerations None.

Choice

Description A pop-up menu of choices. Only one item can be selected at a time.

Inheritance Choice -> Component

Properties Figure 6.9 and Table 6.8 show the properties specific for Choice. The Choice property inspector is a combination of these properties and Component.

Figure 6.9 Properties for Choice

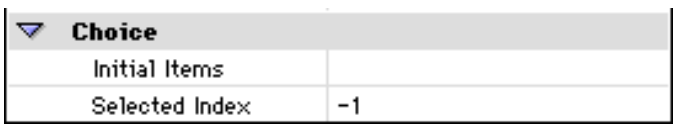


Table 6.8 Choice properties

Item	Specifies
Initial Items	The items that menu contains when it's first created.
Selected Index	The index of the item selected by default. By default, it's -1, which means nothing is selected.

Special considerations To add items to the **Initial Text** property, click the words **Initial Text** and choose **New Menu Item Text** from the **Edit** menu. Enter the text in the new property that ap-

pears. Keep choosing **New Menu Item Text** from the **Edit** menu for each item.

Label

Description A single line of read-only text.

To display a line of text the user can edit, use TextField. To display longer segments of text, use TextArea.

Inheritance Label -> Component

Properties Figure 6.10 and Table 6.9 show the properties specific for Label. The Label property inspector is a combination of these properties and Component.

Figure 6.10 Properties for Label

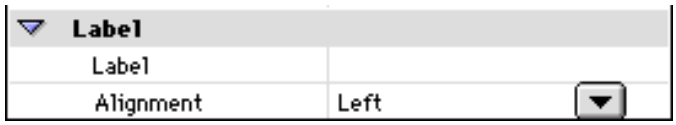


Table 6.9 Label properties

Item	Specifies
Label	The text to display.
Alignment	How to align the text. You can choose from left-aligned, right-aligned, or centered.

Special considerations None.

List

Description A scrolling list of strings. You can select one or more items.

Inheritance List -> Component

Properties Figure 6.11 and Table 6.10 show the properties specific for List. The List property inspector is a combination of these properties and Component.

Figure 6.11 Properties for List

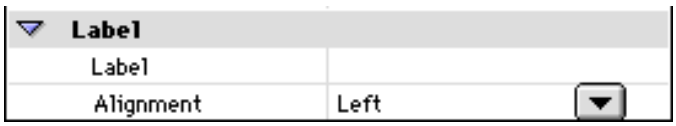


Table 6.10 List properties

Item	Specifies
Initial Items	The items that the list contains when it's first created.
Multiple Selection	Whether you can select more than one item.
Visible Index	The index of the item that the list scrolls to when you first create the list. By default, it's -1, which means the beginning of the list is displayed.

Special considerations To add items to the **Initial Items** property, click the words **Initial Items** and choose **New Text Item** from the **Edit** menu. Enter the text in the new property that appears. Keep choosing **New Text Item** from the **Edit** menu for each item.

Scrollbar

Description A scroll bar that lets the user select from a range of values. A scroll bar has a minimum and maximum value and the thumb represents the selected value.

Inheritance Scrollbar -> Component

Properties Figure 6.12 and Table 6.11 show the properties specific for Scrollbar. The Scrollbar property inspector is a combination of these properties and Component.

Figure 6.12 Properties for Scrollbar

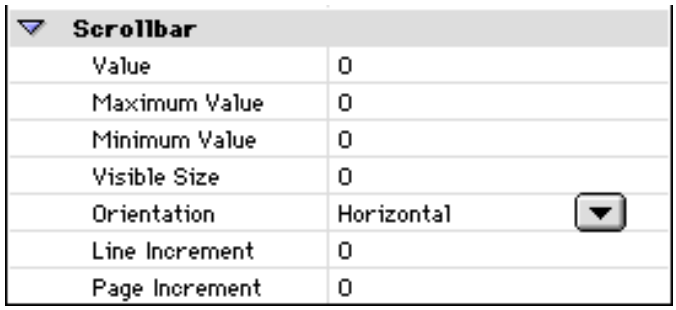


Table 6.11 Scrollbar properties

Item	Specifies
Value	The initial value displayed.
Maximum Value	The initial maximum value.
Minimum Value	The initial minimum value.
Visible Size	The range of values that the width of the scroll bar's thumb represents. On the Mac OS, this property is ignored, since the size of the thumb does not change.
Orientation	Whether the scroll bar is horizontal or vertical.
Line Increment	The amount that AWT adds to or subtracts from the scroll bar's value when you scroll line-by-line. In the Mac OS, you scroll line-by-line when you click the arrows at either end of the scroll bar.
Page Increment	The amount that AWT adds to or subtracts from the scroll bar' value when you scroll page-by-page. In the Mac OS, you scroll page-by-page when you click in the scroll bar outside of the thumb.

Special considerations None.

TextComponent

Description A base class for editable text objects. You cannot add a TextComponent to a JPob, but all TextField and TextArea are derived from it.

Inheritance TextComponent-> Component

Properties Figure 6.15 and Table 6.14 show the properties specific for TextComponent.

Figure 6.13 Properties for TextComponent

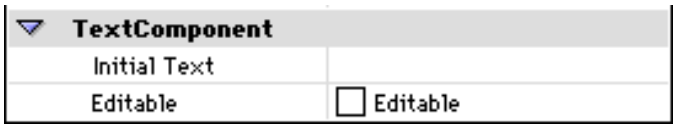


Table 6.12 TextComponent properties

Item	Specifies
Initial Text	The text displayed in the component.
Editable	Whether the user can edit the text.

Special considerations None.

TextArea

Description A multiple-line block of text you can edit.

Inheritance TextArea -> TextComponent-> Component

Properties Figure 6.14 and Table 6.13 show the properties specific for TextArea.

Figure 6.14 Properties for TextArea

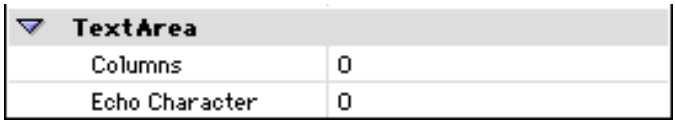


Table 6.13 TextArea properties

Item	Specifies
Rows	The number of lines of text in the component.
Columns	The number of columns wide the component is.

Special considerations None.

TextField

Description A single line of text you can edit.

Inheritance TextField -> TextComponent-> Component

Properties Figure 6.15 and Table 6.14 show the properties specific for TextField.

Figure 6.15 Properties for TextField

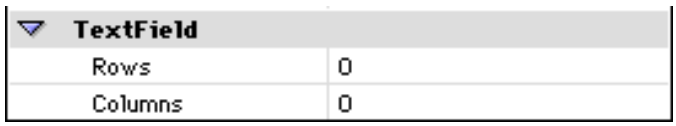


Table 6.14 TextField properties

Item	Specifies
Columns	The number of columns width the component is.
Echo Character	The ASCII value of the character to echo when the user enters text in the field. Use this if the field is for sensitive information, such as a password. If you want the field to echo what the user typed, enter 0.

7

Constructor Menu Reference

This chapter lists each menu item in Constructor and describes its purpose.

Constructor Menu Reference Overview

The Constructor menus include:

- Apple Menu
- Edit Menu
- Arrange Menu
- File Menu
- Window Menu
- Layout Menu

The **Apple**, **File**, **Edit**, and **Windows** menus are visible at all times. The **Arrange** and **Layout** menus appear when a layout editor is active.

Apple Menu

The **Apple** menu contains one Constructor-related item.

About Constructor…

Choose this item to see the way-cool About Box, and the credits for the fine people who put Constructor together for you.

File Menu

Use the **File** menu commands for managing Constructor project files. The Constructor **File** menu has all the usual file-related commands you are accustomed to seeing in a Mac OS application.

New Project File

Creates a new Constructor project file.

See "Using Constructor Files."

Open Project File…

Uses the *Standard Get File* dialog to open an existing Constructor project file.

See "Using Constructor Files."

Close

Closes the active window. If you close the project window and there are unsaved changes, you will be given the chance to save changes.

This item is enabled when there is a window open.

See "Using Constructor Files."

Save

Saves an existing project file. If no file exists, this command displays the *Standard Put File* dialog so you can specify the folder where you want to store the file. By convention, Constructor project file names end with .jpob.

This item is enabled when there is a project open that has been changed without being saved.

See "Using Constructor Files."

Save As…

Save a file to a new name. This command displays the *Standard Put File* dialog so you can specify the folder in which you want to store the file. By convention, Constructor project files end with .jpob.

This item is enabled when there is a project open.

See "Using Constructor Files."

Generate Source…

Displays a dialog that lets you choose the name for the Java source file that Constructor automatically generates.

See "Using Constructor Files."

Page Setup…

Displays the standard dialog for choosing page setup options such as paper size and page orientation.

Print…

Prints the active Constructor window.

This item is enabled when there is a document open.

Quit

Quits Constructor. You will be asked to confirm this operation if any open files have been changed without saving.

Edit Menu

In addition to the standard **Edit** menu commands for copy and paste, the Constructor **Edit** menu contains commands that allow you to create and modify layouts and components.

The menu item text usually includes a word such as "item," "layout," "menu," or other term to clearly reflect the nature of the item

you are acting upon. In some cases, there may be two or more different kinds of items listed. For example, when editing a menu, the **Edit** menu has a few commands to create new items related to menus, including **New Menu** , **New Menu Item** , and **New Separator Item** .

Menu items are enabled and disabled as appropriate.

Undo/Redo/Can't Undo Action

The actual text varies depending upon the most recent action. Undo/Redo toggle so that you can repeatedly switch between undoing and redoing a single action.

Constructor supports single-level undo. Once you perform a new action, you lose the ability to undo or redo the previous action.

Cut Item

Removes the selected items from the active window and puts them on the clipboard.

Copy Item

Makes a copy of the selected items in the active window and puts them on the clipboard.

Paste Item

Makes a copy of items in the clipboard and puts them in the active window.

Clear Item

Deletes selected items from the active window.

Select All

Selects all items in the active window.

Duplicate Item

Makes a duplicate of the selected items in the active window.

New Item

Creates a new item. The menu text reflects the exact nature of the item. It might be a layout or a new item in a Choice or List.

This item is disabled for most components. To create a new component, drag the correct one from the Display Classes window.

Edit Item

This command opens a component editor. This could be the layout window or a property inspector. See "Opening a Component for Editing."

When an editor is already open and an item selected in the editor, this command opens the property inspector window for the item.

Arrange Menu

The **Arrange** menu appears when a layout editor is active. This menu provides tools for aligning and distributing objects in the layout.

Arrange Objects…

Open the Arrange Objects dialog.

See "Aligning components," "Distributing components."

Align Left Edges

Aligns selected objects along the left of the leftmost selected object.

See "Aligning components."

Align Horizontal Centers

Aligns selected objects halfway between the left of the leftmost selected object and the right of the rightmost selected object.

See "Aligning components."

Align Right Edges

Aligns selected objects along the right of the rightmost selected object.

See "Aligning components."

Spread Horizontally

Distributes selected objects evenly from the left of the leftmost selected object to the right of the rightmost selected object.

See "Distributing components."

Spread Horizontally in Container

Distributes selected objects evenly from the left edge to the right edge of the container that holds all the objects.

See "Distributing components."

Align Top Edges

Aligns selected objects along the top of the topmost selected object.

See "Aligning components."

Align Vertical Centers

Aligns selected objects halfway between the top of the topmost selected object and the bottom of the bottommost selected object.

See "Aligning components."

Align Bottom Edges

Aligns selected objects along the bottom of the bottommost selected object.

See "Aligning components."

Spread Vertically

Distributes selected objects evenly from the top of the topmost selected object to the bottom of the bottommost selected object.

See "Distributing components."

Spread Vertically in Container

Distributes selected objects from the top edge to the bottom edge of the container that holds all the objects.

See "Distributing components."

Layout Menu

The **Layout** menu is visible when a layout editor is active. Items in this menu control the layout grid, and object visibility. Most items in the menu are enabled at all times.

Show/Hide Grid

Shows or hides a dotted rectangular grid, like graph paper, to help you align and arrange your objects. You can specify the size of the grid and whether to "snap" the objects to the grid.

See "Using a grid."

Snap/Don't Snap To Grid

Toggles the snap-to-grid setting. When this setting is on, dragging and dropping a object causes it to snap to the nearest grid line.

See "Using a grid."

Edit Grid

Displays the Grid dialog that allows you to specify how large to make the grid rectangles (in pixels).

See "Using a grid."

Show/Hide Object IDs

Show or hide component names in the upper right corner of the object.

See "Displaying component name."

Show/Hide Object Edges

Show or hide the edges of objects which are otherwise invisible, such as Canvas and Panel. Showing the edge also shows the size of unboxed objects such as Label and Checkbox, making it easier to adjust them and avoid overlaps.

See "Displaying component edges."

Show/Hide Invisible Objects

Show or hide objects that have the *Visible* check box in the property inspector deselected.

See "Displaying invisible components."

Show Hierarchy

If the Hierarchy window is not open, this item opens the window. If the window is open but inactive, this item activates the window. To close the Hierarchy window, click the window's close box.

This item is disabled if the Hierarchy window is open and active.

See "Hierarchy Window."

Window Menu

The **Window** menu contains items that display or activate various Constructor windows. This menu is always available, but certain items may be disabled when appropriate.

Property Inspector

This command opens a property inspector window related to the item.

Catalog

If the Display Classes window is not open, choosing this item opens the window. If the window is open but inactive, this item activates the Display Classes window. To close the Display Classes window, click in the window's close box.

This item is disabled if the Display Classes window is open and active.

See "Catalog window."

Alignment Palette

Show the alignment palette. To hide the palette, click in the palette's close box.

See "Aligning components," "Distributing components."

Zoom Window

Zooms the active window between it's default size and full screen size.

Index

Symbols

.jpob file 14

A

aligning components 55
alignment palette
    for aligning components 56
    for distributing components 60
Apple menu, Constructor 103
Arrange menu, Constructor 107
Arrange Objects dialog
    for aligning components 58
    for distributing components 61

B

Button properties 93

C

Canvas properties 94
Checkbox properties 94
CheckboxGroup 94
Choice properties 95
classes
    in Constructor 13
component
    adding to a layout 47
    aligning 55
    arranging hierarchy 63
    arranging visually 52–63
    copying 35
    customizing 67
    deleting 37, 66
    displaying edges 50
    displaying features 49–52
    displaying invisible 51
    displaying name 49
    distributing 59
    distributing by gap 62
    distributing in container 60
    duplicate name 36
    make a layout 33
    make a menu bar 35
    making a new 31–35
    moving 53
    name 36
    nudging 54
    opening editor 36
    property inspector window 36, 45
    removing 66
    resizing 62
    selecting obscured 52
    using grid to align 54
Component properties 87
component property inspector
    structure 84
    using 84
Container properties 90
containment hierarchy 63
    modifying 64
copying items 22
customizing component types 67

D

deleting items 23
Dialog properties 92
Display Classes window 40
display classes window 40
distributing components 59
    by specified gap 62
    in container 60
dragging items 22
duplicate component name 36
duplicating items 23

E

Edit menu, Constructor 105
edit mode for text 79
editing text items 24, 79
editor window
    for layouts 40–46
    for menus 69–73
    opening 36

F

File menu, Constructor 104
file privileges 19
Frame properties 91

G

Generate Source… command 15
grid 54

H

hierarchy window 43
    compared to layout window 45
hierarchy, containment for components 63

I

installing Constructor 9

J

Java Package field 15
Java source files 14
.jpob file 14

L

Label properties 96
layout
    adding components 47
    create component 33
    editing 47–67
    reanimating 25
layout editor
    windows 40–46
Layout menu, Constructor 109
layout window 42
    compared to hierarchy window 45
    grid in 54
Layout windows 42
List properties 96
lists
    in Constructor 24
    opening and closing 24
    selecting items in 24
`Locate` 29
`Locate()` 26

M

marquee selection 21
menu
    adding a submenu 82
    changing item order 81
    create 35
    deleting an item 82
    editing items 77–82
    editor window 71
    editor windows 69–73
    making a new item 79
    making new 74
    modifiying an item 80
    property window 72
    removing an item 82
    removing from menu bar 77
    selecting menu item 78
menu bar
    adding existing menu 74
    adding menu 74
    changing menu order 76
    creating 35
    deleting a menu 77
    editing 73–77
    editor window 70
    editor See also menu editor
    removing a menu 77
menu item property window 72, 80
menu item See menu
moving items in Constructor 22

P

Panel properties 90
PPob
    setting top-level view 34
project files 14
project window 18
property inspector windows 36

R

read/write permission 19
`Reanimate` 27
`Reanimate()` 25
`ReanimateMenu` 28
`ReanimateMenuBar` 29
`ReanimateWindow` 28
reanimating 25
Reanimator 25
`Reanimator` 25
redo in Constructor 25

resizing components 62

S

Scrollbar properties 97
selecting
    items in Constructor 21
    menu item 78
    obscured component 52
Source File Name field 15
source files 14
system requirements 7

T

text editing 24
TextArea properties 99
TextComponent properties 99
TextField properties 100
top-level view 34

U

undo in Constructor 25
user interface features 21

V

visual interface, building 47–67

W

window hierarchy 43
Window properties 91
window, Display Classes 40
Windows menu, Constructor 111
windows, layout 42

CodeWarrior

Constructor for Java

Credits

writing lead:	John Spicer
other writers:	Jeff Mattson and Jim Trudeau
engineering:	Andrew Southwick, Eric Scouten, Clinton Popetz, Robin Mair, and Eric Shapiro
frontline warriors:	Clint Popetz, Eric Scouten, Michael Little, David Mark, Perdo Larios, and Ferry Sutanto

Guide to CodeWarrior Documentation

CodeWarrior documentation is modular, like the underlying tools. There are manuals for the core tools, languages, libraries, and targets. The exact documentation provided with any CodeWarrior product is tailored to the tools included with the product. Your product will not have every manual listed here. However, you will probably have additional manuals (not listed here) for utilities or other software specific to your product.

Core Documentation	
IDE User Guide	How to use the CodeWarrior IDE
Debugger User Guide	How to use the CodeWarrior debugger
CodeWarrior Core Tutorials	Step-by-step introduction to IDE components
Language/Compiler Documentation	
C Compilers Reference	Information on the C and C++ compilers
Pascal Compilers Reference	Information on the Pascal and Object Pascal compilers
Error Reference	Comprehensive list of compiler/linker error messages, with many fixes
Pascal Language Reference	The Metrowerks implementation of ANS Pascal
Assembler Guide	Stand-alone assembler manual
Command-Line Tools Reference	Command-line options for Mac OS and Be compilers
Plugin API Manual	The CodeWarrior plugin compiler/linker API
Library Documentation	
MSL C Reference	Function reference for the Metrowerks standard C library
MSL C++ Reference	Function reference for the Metrowerks standard C++ library
Pascal Library Reference	Function reference for the Metrowerks ANS Pascal library
The PowerPlant Book	Guide to the Metrowerks application framework for Mac OS
PowerPlant Advanced Topics	Advanced topics in PowerPlant programming for Mac OS
MFC Reference	Reference for the Microsoft Foundation Classes for Win32
Win32 SDK Reference	Microsoft's Reference for the Win32 API
Targeting Manuals	
Targeting BeOS	How to use CodeWarrior to program for BeOS
Targeting the Java VM	How to use CodeWarrior to program for the Java virtual machine
Targeting Mac OS	How to use CodeWarrior to program for Mac OS
Targeting MIPS	How to use CodeWarrior to program for MIPS embedded processors
Targeting Palm OS	How to use CodeWarrior to program for PalmPilot
Targeting PlayStation OS	How to use CodeWarrior to program for the PlayStation game console
Targeting PowerPC Embedded Systems	How to use CodeWarrior to program for PPC embedded processors
Targeting Win32	How to use CodeWarrior to program for Windows 95/NT